Motion Drive

Variateur numérique pour moteur Brushless
Série IMD

Manuel d'utilisation

Lire attentivement ce manuel avant la mise en route et respecter toutes les indications avec le symbole :

SERAD SA
271, route des crêtes
44440 TEILLE – France
☎ +33 (0)2 40 97 24 54
📠 +33 (0)2 40 97 27 04
💻 http://www.serad.fr
✉ info@serad.fr

Nous nous réservons le droit de modifier sans préavis tout ou partie des caractéristiques de nos appareils

Sommaire

1- INTRODUCTION 8

1-1- MISE EN GARDE 8
1-2- DESCRIPTION DU VARIATEUR IMD 9
1-2-1- Généralités : *9*
1-2-2- Données techniques: *9*
1-3- DESCRIPTION DU LOGICIEL IDPL 12
1-3-1- Généralités : *12*
1-3-2- Données techniques : *12*
1-3-3- Langage de programmation iDPL : *12*

2- INSTALLATION 13

2-1- GENERALITES 13
2-2- VUE DE FACE 15
2-3- VUE DE DESSUS 16
2-4- VUE DE DESSOUS 17
2-5- MONTAGE 18
2-6- AFFECTATION ET BROCHAGES DES CONNECTEURS 19
2-7- CABLES 28
2-8- SCHEMAS DE RACCORDEMENT / PROTECTION : 30
2-9- VARIATEUR AUTONOME 31
2-10- VARIATEUR PILOTE PAR UNE COMMANDE D'AXE 32
2-11- RACCORDEMENT D'UN FREIN MOTEUR 33
2-12- VERIFICATIONS AVANT MISE EN ROUTE 33

3- LOGICIEL IDPL 34

3-1- INSTALLATION DU LOGICIEL IDPL 34
3-1-1- Configuration du système *34*
3-1-2- Procédure d'installation du logiciel iDPL *35*
3-1-3- Les répertoires *36*
3-2- PRESENTATION 36
3-2-1- Les modes d'utilisation *36*
3-2-2- Ecran initial *39*
3-2-3- Gestion d'un projet *42*
3-2-4- Contenu d'un projet *46*
3-3- MENUS ET ICONES 47
3-3-1- Projet *47*
3-3-2- Paramètres *49*
3-3-3- Communication *69*
3-3-4- Outils de réglages *73*
3-3-5- Motion control *81*
3-3-6- Langage iDPL *86*
3-3-7- Options *88*
3-3-8- Aide *90*

4- REGLAGE DU VARIATEUR 91

4-1- REGLAGE DES PARAMETRES MOTEUR ET RETOUR POSITION 91
4-2- REGLAGE DU MODE DE DEVERROUILLAGE VARIATEUR 93
4-3- LES MODES DE FONCTIONNEMENT 94
4-4- REGLAGE AUTOMATIQUE DES BOUCLES DE REGULATION 95
4-4-1- Auto tuning de la boucle de courant : *95*
4-4-2- Auto tuning de la boucle de vitesse : *95*
4-4-3- Auto tuning de la boucle de position : *95*
4-4-4- Auto tuning complet : *95*
4-4-5- Mise en garde sur l'auto tuning : *96*
4-5- REGLAGE MANUEL DES BOUCLES DE REGULATION 96
4-5-1- Réglage de la boucle de courant *96*

4-5-2- Réglage de la boucle de vitesse *99*
4-5-3- Réglage de la boucle de position *102*
4-6- AUTRES REGLAGES 106
4-6-1- Réglage en boucle de vitesse *106*
4-6-2- Réglage en double boucle résolveur/codeur *107*
4-6-3- Réglage en entrée stepper *107*

5- LES TRAJECTOIRES 108

5-1- INTRODUCTION : 108
5-2- TRAJECTOIRES PAR CARTE I/O 109
5-2-1- Fonctionnement avec carte I/O: *109*
5-2-2- Mise en oeuvre avec carte I/O: *110*
5-3- TRAJECTOIRES PAR BUS DE COMMUNICATION 113
5-3-1- Fonctionnement par bus de communication: *113*
5-3-2- Mise en oeuvre par bus de communication: *114*
5-4- TRAJECTOIRES PAR CARTE I/O EN MODE AVANCE : 118
5-4-1- Fonctionnemen tavec carte I/O en mode avancé : *118*
5-4-2- Mise en oeuvre avec carte I/O en mode avancé : *122*

6- LANGAGE DE PROGRAMMATION IDPL 124

6-1- INTRODUCTION 124
6-1-1- Introduction *124*
6-1-2- Affectation du plan mémoire *124*
6-2- LES VARIABLES 126
6-2-1- Variables *126*
6-2-2- Conversions de type de variables *127*
6-2-3- Notation numériques *128*
6-2-4- Variables globales sauvegardées *128*
6-3- LES DONNEES SAUVEGARDEES 129
6-4- LES PARAMETRES 132
6-5- LES TACHES 134
6-5-1- Principes du multitâches *134*
6-5-2- Priorité des tâches *134*
6-5-3- Gestion des tâches *134*
6-5-4- Structure d'une tâche basic *136*

7- PROGRAMMATION DU CONTROLE DE MOUVEMENT 142

7-1- INTRODUCTION 142
7-2- PARAMETRAGE D'UN AXE 142
7-2-1- Réglage d'un axe *142*
7-2-2- Unité utilisateur *144*
7-2-3- Profil de vitesse *145*
7-3- MODE ASSERVI / NON ASSERVI 145
7-3-1- Passage en mode non asservi *145*
7-3-2- Passage en mode asservi *146*
7-4- PRISE D'ORIGINE 146
7-4-1- Définition : *146*
7-4-2- Configuration de la POM sous DPL : *147*
7-4-3- Les types de POM : *147*
7-5- DECLARATION D'UN AXE EN MODE VIRTUEL 152
7-6- POSITIONNEMENT 153
7-6-1- Mouvements absolus *153*
7-6-2- Mouvements relatifs *154*
7-6-3- Mouvements infinis *155*
7-6-4- Arrêt d'un mouvement *156*
7-6-5- Positionnement par bus de communication *156*
7-7- SYNCHRONISATION 158
7-7-1- Arbre électrique : *158*
7-7-2- Mouvements synchronisés *160*
7-7-3- Mouvement de correction *165*

7-7-4- Came ... *167*
7-7-5- Multiaxes par CANopen ... *176*
7-7-6- Débrayage d'un mouvement synchronisé ... *178*
7-8- CAPTURE ... 180
7-8-1- Capture : ... *180*
7-8-2- Recalage automatique ... *181*
7-9- MOUVEMENTS DECLENCHES ... 183
7-10- MAITRE VIRTUEL ... 185

8- PROGRAMMATION DE L'AUTOMATE ... 188

8-1- ENTREES/SORTIES LOGIQUES ... 188
8-1-1- Lecture des entrées ... *188*
8-1-2- Ecriture des sorties ... *188*
8-1-3- Lecture des sorties ... *189*
8-1-4- Attente d'un état ... *189*
8-1-5- Test d'un état ... *190*
8-2- ENTREES/SORTIES ANALOGIQUES ... 190
8-2-1- Lecture d'une entrée ... *190*
8-2-2- Ecriture d'une sortie ... *190*
8-3- TEMPORISATIONS ... 191
8-3-1- Attente passive ... *191*
8-3-2- Attente active ... *191*
8-4- COMPTEURS ... 193
8-4-1- Compteurs ... *193*
8-5- BOITE A CAMES ... 195

9- LISTE DES OPERATEURS ET INSTRUCTIONS ... 197

9-1- PROGRAMME ... 197
9-2- ARITHMETIQUE ... 197
9-3- MATHEMATIQUE ... 197
9-4- LOGIQUE ... 198
9-5- TEST ... 198
9-6- CONTROLE DE MOUVEMENT ... 199
9-6-2- Mouvements déclenchés ... *201*
9-6-3- Maître virtuel ... *201*
9-7- AUTOMATE ... 202
9-8- GESTION DES TACHES ... 203
9-9- FLASH, SECURITE, DIVERS ... 203
9-10- LISTE APLHABETIQUE ... 204
9-10-1- Addition (+) ... *204*
9-10-2- Soustraction (-) ... *204*
9-10-3- Multiplication (*) ... *204*
9-10-4- Division (/) ... *205*
9-10-5- Inférieur (<) ... *205*
9-10-6- Inférieur ou égal (<=) ... *206*
9-10-7- Décalage à gauche (<<) ... *206*
9-10-8- Différent (<>) ... *206*
9-10-9- Affectation/Egalité (=) ... *207*
9-10-10- Supérieur (>) ... *207*
9-10-11- Supérieur ou égal (>=) ... *207*
9-10-12- Décalage à droite (>>) ... *208*
9-10-13- ACC - Accélération ... *208*
9-10-14- ADC(1) – Entrée analogique 1 ... *209*
9-10-15- ADC(2) – Entrée analogique 2 ... *209*
9-10-16- ACC% - Accélération en pourcentage ... *209*
9-10-17- AND – Opérateur ET ... *210*
9-10-18- ARCCOS – Cosinus inverse ... *210*
9-10-19- ARCSIN – Sinus inverse ... *210*
9-10-20- ARCTAN – Tangente inverse ... *211*
9-10-21- AXIS – Contrôle la boucle d'asservissement ... *211*

9-10-22- AXIS_S – Lit l'état de la boucle d'asservissement ... *212*
9-10-23- BUFMOV_S ... *212*
9-10-24- CALL – Appel d'un sous-programme ... *212*
9-10-25- CAMBOX – Boîte à cames ... *213*
9-10-26- CAMBOXSEG – Segment de boîte à cames ... *213*
9-10-27- CAMNUM_S – Numéro de la came en cours d'exécution ... *214*
9-10-28- CAMREADPOINT – Position de l'esclave dans la came ... *214*
9-10-29- CAMSEG_S – Numéro d'équation de la came en cours d'exécution ... *214*
9-10-30- CAPTURE1 ... *215*
9-10-31- CLEAR – Met à zéro la position de l'axe ... *216*
9-10-32- CLEARMASTER - met à zéro la position du codeur maître ... *216*
9-10-33- COMCOUNTER – Retourne le nombre de trames échangées ... *216*
9-10-34- CONTINUE – Continue l'exécution d'une tâche ... *217*
9-10-35- COS - Cosinus ... *217*
9-10-36- COUNTER - Initialise le compteur à une valeur ... *218*
9-10-37- COUNTER_S – Renvoie la valeur d'un compteur ... *218*
9-10-38- DAC - Sortie analogique ... *218*
9-10-39- DEC - Décélération ... *219*
9-10-40- DEC% - Décélération en pourcentage ... *219*
9-10-41- DELAY – Attente passive ... *219*
9-10-42- DISABLERECALE– Désactivation du recalage ... *220*
9-10-43- DISPLAY – Afficheur 7 segments ... *220*
9-10-44- ENABLERECALE– Fonction de recalage automatique sur capture ... *220*
9-10-45- EXIT SUB – Sortie d'un sous-programme ... *221*
9-10-46- ENDCAM – Arrêt d'une came ... *222*
9-10-47- EXP - Exponentiel ... *222*
9-10-48- FEMAX_S – Limite d'erreur de poursuite ... *222*
9-10-49- FE_S - Erreur de poursuite ... *223*
9-10-50- FILTERMASTER – Permet d'appliquer un filtrage lors de mouvement synchrone ... *223*
9-10-51- FRAC – Partie fractionnelle ... *224*
9-10-52- GEARBOX ... *224*
9-10-53- GEARBOXRATIO ... *225*
9-10-54- GOTO – Saut à une étiquette ... *225*
9-10-55- HALT – Arrêter une tâche ... *225*
9-10-56- HOME – Prise d'origine ... *226*
9-10-57- HOME_S – Etat de la prise d'origine ... *227*
9-10-58- HOMEMASTER– Prise d'origine sur le maître ... *227*
9-10-59- HOMEMASTER_S – Etat de la prise d'origine du maître ... *228*
9-10-60- ICORRECTION– fonction de compensation ... *229*
9-10-61- ICORRECTION_S– Etat de la compensation ... *229*
9-10-62- IF - IF ... *229*
9-10-63- INP – Lecture d'une entrée TOR ... *230*
9-10-64- INPB – Lecture d'un bloc 8 entrées ... *230*
9-10-65- INPW – Lecture des 16 entrées logiques ... *231*
9-10-66- INT – Partie entière ... *231*
9-10-67- LOADCAM – Permet de charge une came ... *231*
9-10-68- LOADCAMPOINT – Modification de points d'une came ... *232*
9-10-69- LOADPARAM – Permet de recharger les paramètres du variateur ... *233*
9-10-70- LOADVARIABLE - Permet de transférer les variables sauvegardés ... *233*
9-10-71- LOADTIMER - Charge une temporisation dans une variable ... *233*
9-10-72- LOG - Logarithme ... *233*
9-10-73- LOOP – Mode virtuel ... *234*
9-10-74- MASTEROFFSET – Décale dynamiquement la position du maître ... *234*
9-10-75- MERGE – définit l'enchaînement ... *234*
9-10-76- MOD - Modulo ... *235*
9-10-77- MOVA – Mouvement absolu ... *235*
9-10-78- MOVE_S – Etat du mouvement ... *235*
9-10-79- MOVEMASTER_S – Indique si un mouvement est en cours lorsqu'on est en maître virtuel ... *236*
9-10-80- MOVR – Mouvement relatif ... *236*
9-10-81- MOVS - permet d'effectuer une synchronisation entre un axe esclave et un maître. ... *237*

9-10-82- NEXTTASK 237
9-10-83- NOT – Opérateur complément 237
9-10-84- OR - Opérateur ou 238
9-10-85- ORDER – Numéro d'ordre du mouvement 238
9-10-86- ORDER_S – Numéro d'ordre courant 238
9-10-87- OUT – Ecriture d'une sortie 239
9-10-88- OUTB – Ecriture d'un bloc de 8 sorties 239
9-10-89- POS – Position à atteindre 239
9-10-90- POS_S – Position réelle 240
9-10-91- POSMASTER_S – Position réelle du maître 240
9-10-92- PROG .. END PROG – Début d'un programme 241
9-10-93- READCAM – Permet de lire un point de came 241
9-10-94- READI - Lecture d'un entier en FRAM 241
9-10-95- READL -Lecture d'un entier long en FRAM 242
9-10-96- READR - Lecture d'un réel en FRAM 242
9-10-97- READPARAM - Lecture d'un paramètre 242
9-10-98- REG1 S 242
9-10-99- REGPOS1 S 243
9-10-100- REPEAT ... UNTIL – Répétition d'une boucle 243
9-10-101- RESTART – Redémarrage du système 244
9-10-102- RUN – Lance une tâche 244
9-10-103- SAVEPARAM - Permet de sauvegarder les paramètres du variateur 244
9-10-104- SAVEVARIABLE – Permet de sauvegarder les variables 245
9-10-105- SECURITY – Définit les actions de sécurités 245
9-10-106- SETUPCOUNTER – Configure un compteur 246
9-10-107- SGN - Signe 246
9-10-108- SIN - Sinus 246
9-10-109- SLAVEOFFSET – Décale dynamiquement la position de l'esclave 246
9-10-110- SQR - Racine carrée 247
9-10-111- SSTOP – Arrêt d'un axe 247
9-10-112- SSTOPMASTER – Arrête le mouvement du maître virtuel sans attente 247
9-10-113- STARTCAM – Exécute une came 248
9-10-114- STARTCAMBOX – Lance une boîte à cames 248
9-10-115- STARTGEARBOX - Lance l'arbre électrique 248
9-10-116- STATUS – Etat d'une tâche 249
9-10-117- STOP - Arrêt d'un axe 249
9-10-118- STOPCAMBOX – Arrête une boîte à cames 250
9-10-119- STOPS - permet d'arrêter l'instruction MOVS 250
9-10-120- STOPS_S - état du mouvement synchronisé 250
9-10-121- STOPMASTER – Arrête le mouvement du maître virtuel 251
9-10-122- STTA – Lance un mouvement absolu 251
9-10-123- STTI – Lance un mouvement infini 252
9-10-124- STTR – Lance un mouvement relatif 252
9-10-125- SUB .. END SUB – Sous-programme 252
9-10-126- SUSPEND – Suspend une tâche 253
9-10-127- TAN - Tangente 253
9-10-128- TIME - Base de temps étendue 253
9-10-129- TIMER – Comparaison une variable à Time 254
9-10-130- TRAJA – Trajectoire absolue 254
9-10-131- TRAJR – Trajectoire relative 255
9-10-132- TRIGGERC - Mouvement déclenché sur entrée capture 255
9-10-133- TRIGGERI – Mouvement déclenché sur entrée 256
9-10-134- TRIGGERP – Mouvement déclenché sur position maître 256
9-10-135- TRIGGERR – Annule le mouvement déclenché 257
9-10-136- TRIGGERS – Active le mouvement déclenché 257
9-10-137- VEL - Vitesse 257
9-10-138- VEL_S 257
9-10-139- VEL% 258
9-10-140- VELMASTER_S 258
9-10-141- VERSION – Version de l'operating system (Firmware) 258

9-10-142- VIRTUALMASTER – Active/désactive le maître virtuel *258*
9-10-143- WAIT - Attente d'une condition *259*
9-10-144- WRITECAM – Permet d'écrire un point de came *259*
9-10-145- WRITEI - Ecriture d'un entier en FRAM *259*
9-10-146- WRITEL - Ecriture d'un entier long en FRAM *259*
9-10-147- WRITEPARAM – Ecriture d'un paramètre *260*
9-10-148- WRITER - Ecriture d'un réel en FRAM *260*
9-10-149- XOR – Opérateur ou exclusif *260*

10- ANNEXES 261

10-1- AFFICHEUR STATUS 7 SEGMENTS 261
10-1-1- Description des messages : *261*
10-1-2- Messages d'erreur : *263*
10-2- CANOPEN 267
10-2-1- Définition *267*
10-2-2- Carte IMDCANI pour drive IMD *272*
10-2-3- Liste des instructions *276*
10-2-4- Exemples *284*
10-3- MODBUS 288
10-3-1- Définition *288*
10-3-2- Dictionnaire *290*
10-4- TELEMAINTENANCE 292
10-4-1- Raccordement *292*
10-4-2- Etablissement de la liaison *293*
10-4-3- Liste des modems validés *301*

1- Introduction

1-1- Mise en garde

Avant la première mise en service de l'installation, veuillez lire les informations suivantes afin d'éviter des dommages corporels et/ou matériels.

Le montage, le raccordement, la mise en service et la maintenance de l'appareil ne peuvent être réalisés que par des personnes qualifiées et doivent obéir aux normes nationales et internationales (DIN, VDE, EN, IEC …). Le non respect de ces normes peut engendrer de graves dommages matériels.

De plus, il est indispensable de respecter les instructions de sécurité. Des blessures et dommages corporels peuvent résulter d'une méconnaissance de ces instructions.

Les règles de prévention des accidents sont les suivantes :

• VDE 0100	Spécification pour l'installation des systèmes de puissance jusqu'à 1000 V
• VDE 0113	Equipement électrique de machines
• VDE 0160	Equipement de système de puissance avec des composants électroniques

- ***Ne jamais ouvrir l'appareil.***
- ***Des hautes tensions pouvant être dangereuses sont appliquées à l'intérieur du variateur et des connecteurs. Pour cela, couper l'alimentation réseau et attendre au moins 5 minutes pour que les condensateurs se déchargent avant de débrancher un connecteur.***
- ***Ne jamais débrancher ou brancher de connecteurs sous tension.***
- ***L'appareil peut comporter des surfaces très chaudes.***

Ne pas manipuler l'appareil de façon inapproprié sous peine de détérioration de certains composants électroniques par décharges électrostatiques.

Toutes les mesures existantes ont été prises afin de garantir l'exactitude et l'intégrité de la documentation présente, toutefois celle-ci peut contenir des erreurs. Aucune responsabilité ne sera assumée par SERAD pour tout dommage causé par l'utilisation du logiciel et de la documentation ci-jointe.

Nous nous réservons le droit de modifier sans préavis tout ou partie des caractéristiques de nos appareils.

1-2- Description du variateur iMD

1-2-1- Généralités :

Les variateurs intelligents brushless série iMD sont tout spécialement adaptés aux dynamiques élevées.

Ils intègrent l'alimentation, la résistance de freinage et le filtre réseau.

Ils peuvent être utilisés en mode couple, en mode vitesse, en mode positionnement.

Les bus de communications MODBUS et CANopen assurent des configurations en réseau.

Grâce à leur langage Basic multitâches, leurs fonctions de MOTION et automate intégrées, ils répondent aux applications les plus diverses.

1-2-2- Données techniques:

Alimentation :	230V à 480V AC ±10% triphasée ou 230V AC ±10% monophasée
Alimentation auxiliaire :	24 V DC ±10% 0,4A typique 0,7A maxi si toutes options
Filtre réseau :	Intégré
Fréquence de découpage :	6.67 KHz, commande sinusoïdale du moteur
Tension DC Bus :	De 310 V à 680V
Résistance de freinage :	Intégrée : 75 ohms 60W Possibilité d'ajouter une résistance externe : Valeur Min. — Puissance Cont. Max. — Puissance Imp. Max 60 Ω — 5 KW — 10 KW
Protections :	Court-circuit entre phases, phase à la terre, sur courant, I^2t Surtension, sous-tension Défaut feedback moteur
Retour moteur :	• Résolveur (résolution 16 bits) Précision absolue résolveur ± 0,7° • Codeur SINCOS HIPERFACE, mono-tour ou multi-tours résolution 8 bits par période (en option)

Codeur maître auxiliaire :	• Incrémental : A, /A, B, /B, Z, /Z Fréquence maxi : 6 MHz • Virtuel • Codeur absolu (SSI) • Codeur SINCOS Hiperface (en option)
Emulation codeur :	Incrémental : A, /A, B, /B, Z, /Z de 4 à 100 000 points par tour
Diagnostic :	Afficheur 7 segments
Communication :	RS 232 MODBUS RTU RS 422 (point à point), RS 485 MODBUS RTU (option) CANopen DS402 (option) PROFIBUS DP * (option) SERCOS * 16MB (option)
Entrées logiques :	4 voies (dont 2 entrées standards et 2 rapides: E3 et E4) 12 voies sur module d'extension optionnel (dont 10 entrées standards et 2 rapides: E15 et E16) type : PNP 24 Vdc, 8mA par voie standard et 15mA par voie rapide niveau logique 0 : de 0 à 5 V niveau logique 1 : de 8 à 30 V
Sorties logiques :	2 voies en standard : S1 : relais, 48 Vdc maxi, 48 Vac maxi, 3 A maxi S2 : statique NPN (collecteur ouvert) 24 Vdc, 100 mA 8 voies sur module d'extension optionnel : type : statique PNP 24 Vdc, 500 mA maxi par voie protection contre les courts-circuits et surchauffe
Entrées analogiques :	2 voies : Tension d'entrée : ±10 V Tension d'entrée maxi: ±12 V Impédance d'entrée : 20 Kohm Résolution : 16 bits sur l'entrée 1 12 bits sur l'entrée 2

Sortie analogique :	1 voie : Tension de sortie : ±10 V Courant de sortie maxi: 5 mA Résolution : 8 bits
Architecture :	Processeur DSP 150 MHz et FPGA 100 000 portes Mémoire FLASH pour stockage des programmes et paramètres Mémoire RAM pour stockage des données Mémoire FRAM pour stockage des variables sauvegardées Noyau temps réel multitâches
Boucles de régulation :	Boucle de courant : 75 µs Boucle de vitesse : 150 µs Boucle de position : 150µs
Modes de fonctionnement :	Mode couple Mode vitesse Mode positionnement Fonctions MOTION (mouvement absolu, relatif, infini) Fonctions MOTION avancées (arbre électrique, boite à cames, synchronisation, profil de cames …)
Température de service :	0 à 40°C
Température de stockage :	-10 à 70°C
Indice de protection :	IP 20
Poids	3,6 Kg

Drive	Courant nominal	Courant crête (2s)	Puissance nominale	Dimensions l x h x p
IMD / 1	**1,25 Aeff**	**2,5 Aeff**	**0,7 kVA**	**72 x 293 x 233**
IMD / 2	**2,5 Aeff**	**5 Aeff**	**1,4 kVA**	**72 x 293 x 233**
IMD / 5	**5 Aeff**	**10 Aeff**	**2,8 kVA**	**72 x 293 x 233**
IMD / 10	**10 Aeff**	**20 Aeff**	**5,6 kVA**	**72 x 293 x 233**
IMD / 20	**20 Aeff**	**40 Aeff**	**11,2 kVA**	**125 x 293 x 233**

1-3- Description du logiciel iDPL

1-3-1- Généralités :

L'atelier logiciel iDPL, grâce à son outil graphique, permet de configurer très facilement le variateur à partir d'un PC.

Sous environnement Windows, il offre une convivialité parfaite, des écrans avec multifenêtrages et une aide complète.

Les fonctions d'auto tuning, générateur de trajectoires et oscilloscope assurent une mise en œuvre rapide et optimale.

1-3-2- Données techniques :

- Configuration de tous les paramètres par groupe: moteur, régulation, codeur, E/S analogiques, E/S logiques, communication, sécurités…
- Affichage des paramètres d'états: vitesses, courants, couples, positions…
- Sauvegarde et impression des paramètres sur PC
- Fonctions d'auto-tuning résolveur
- Générateur de trajectoires: position, accélération, décélération, vitesse
- Oscilloscope numérique multicanaux
- Tableau de bord: axe, entrées / sorties
- Reconnaissance automatique du variateur connecté
- Travail possible en mode non connecté (vérification et édition de paramètres…)
- Aide en ligne pour chaque fenêtre

1-3-3- Langage de programmation iDPL :

Les variateurs de la série iMD intègrent un noyau temps réel multitâches et plus de 1000 variables utilisateurs.

Le langage motion basic iDPL permet à l'utilisateur de développer, tester et sauvegarder ses propres programmes applicatifs.

Les applications peuvent être une combinaison des différents mode de fonctionnement (mode couple, vitesse et/ou positionnement).

Les entrées / sorties sont librement utilisées dans le programme, ainsi que les paramètres et les variables.

2- Installation

2-1- Généralités

Il est très important de respecter les points suivants :

↳ Une mauvaise mise à la terre du variateur peut endommager ses composants électroniques.

↳Le variateur doit être installé verticalement pour assurer un refroidissement naturel par convection.

↳ Il doit être à l'abri de l'humidité, des projections de liquide quelconque, de la poussière.

↳ Les câbles résolveur, moteur, codeur devront être blindés, la tresse étant reliée de chaque côté au châssis.

↳ Le câble consigne analogique devra être blindé, la tresse étant reliée de chaque côté au châssis.

↳ Le câble de liaison série RS 232 variateur / PC devra être blindé, la tresse étant reliée de chaque côté au châssis. Il devra être débranché du variateur lorsqu'il n'est plus utilisé. Tous ses câbles, ainsi que les câbles d'entrées-sorties, devront être séparés et éloignés des circuits de puissance.

↳ Il faut prévoir sur toutes les sorties statiques (Q2 à Q10) des diodes de roue libre sur les charges inductives. Ces diodes doivent être placées le plus près possible de la charge. Les conducteurs d'alimentation et de signaux ne doivent pas être le siège de surtensions.

↳ Les normes de sécurité imposent un réarmement manuel après un arrêt provoqué par :
- une coupure secteur
- un appui sur l'arrêt d'urgence
- un défaut variateur.

↳ Pour tout défaut grave, il est obligatoire de couper l'alimentation de puissance du variateur.

↳ La sortie « drive ready » devra être reliée en série dans la boucle d'arrêt d'urgence.

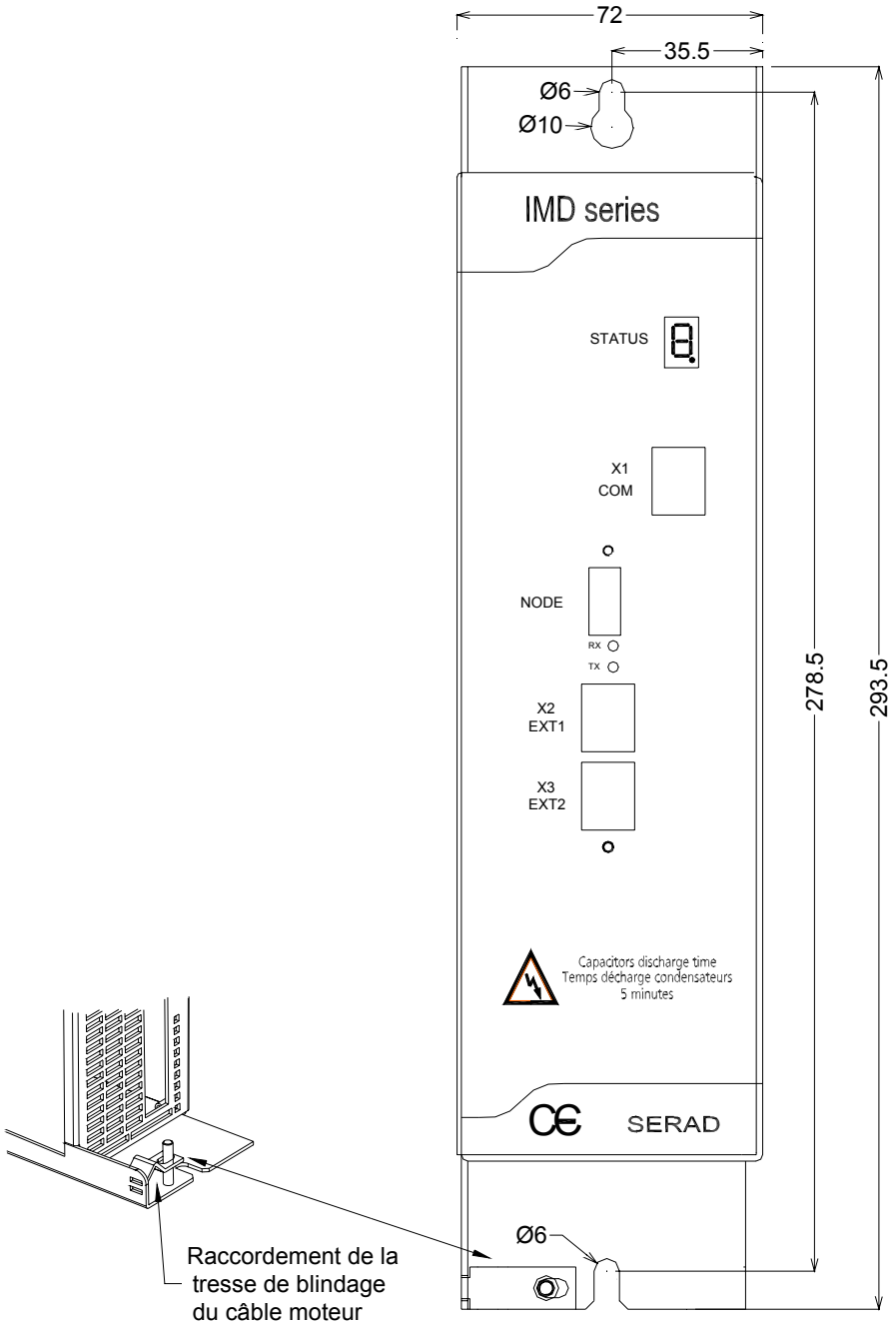
↳ Dans le cas d'un axe fini, les capteurs de fin de course devront être reliés sur les entrées fin de course du variateur ou en série dans la boucle d'arrêt d'urgence

↳ Si le variateur est configuré en mode couple ou vitesse, la validation du variateur se fera à partir de l'entrée ENABLE du variateur et devra être gérée par l'appareil en amont (commande d'axes, automate …)

↳ Si le variateur est configuré en mode position, le paramètre "Erreur de poursuite maxi" devra être réglé.

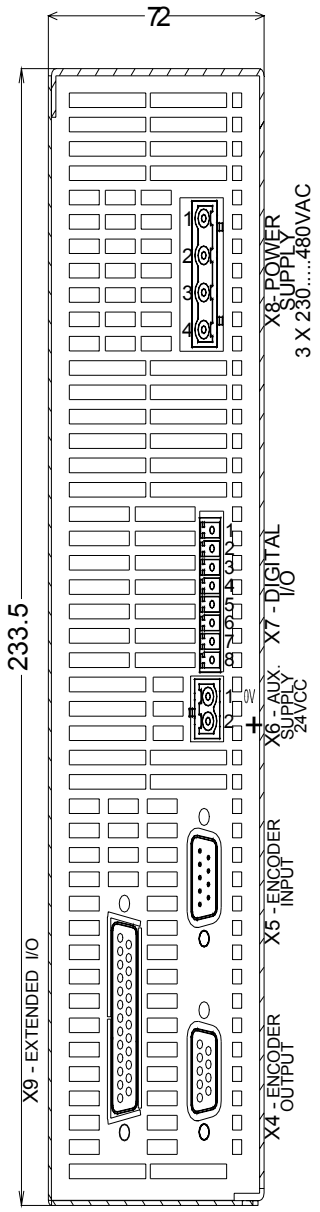
↳ Si le variateur contient un programme applicatif développé à partir du langage iDPL, relier l'information « Puissance armoire électrique OK » sur une entrée automate et la traiter dans une tâche basic non bloquante de sécurité. Sur détection d'une erreur de poursuite, le variateur passe en boucle ouverte et ouvre la sortie « drive ready ».

2-2- Vue de face



	STATUS	Afficheur 7 segments pour diagnostic
X1	COM	Port de communication RS 232 pour paramétrage PC
X2	EXT1	Extension: Bus de communication optionnel
X3	EXT2	Extension: Bus de communication optionnel

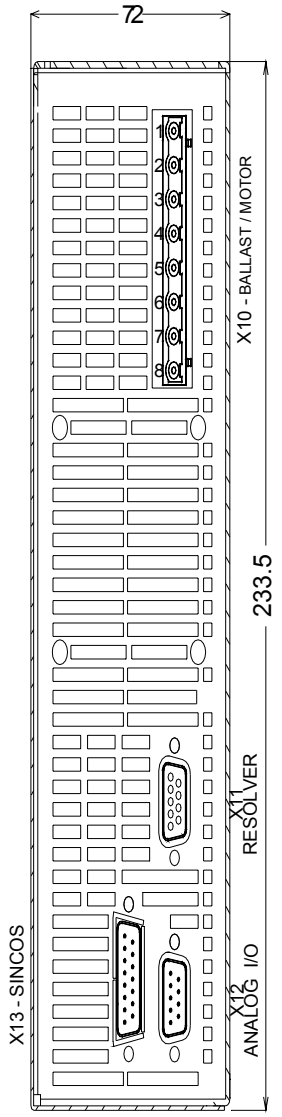
2-3- Vue de dessus



X4	ENCODER OUTPUT	Sortie codeur multifonctions
X5	ENCODER INPUT	Entrée codeur multifonctions
X6	AUX. SUPPLY 24VCC	Alimentation auxiliaire 24 VCC
X7	DIGITAL I/O	Entrées et sorties logiques
X8	POWER SUPPLY	Alimentation monophasée ou triphasée
X9	EXTENDED I/O	Option : Extension d'entrées / sorties logiques

La tension sur le connecteur X8 peut atteindre 480V!

2-4- Vue de dessous



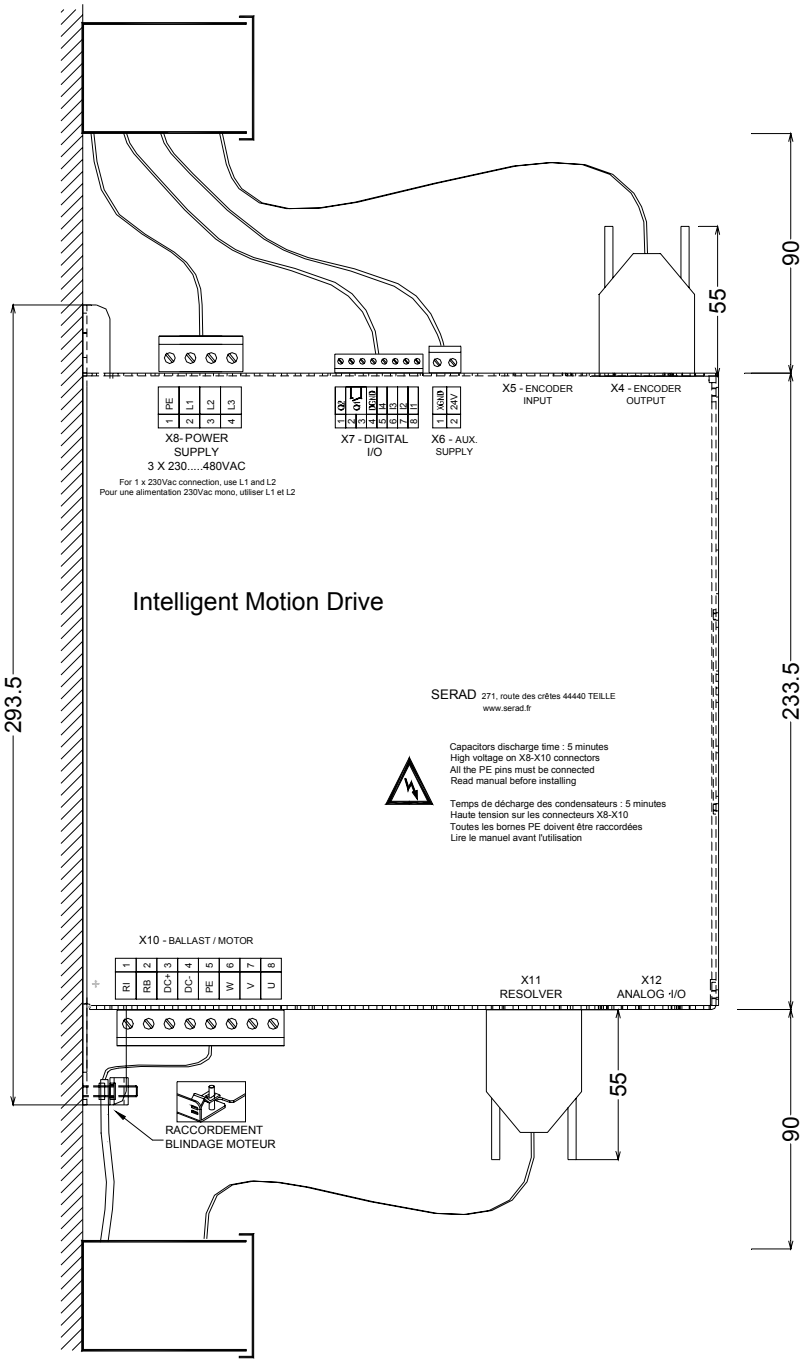
X10	BALLAST / MOTOR	Alimentation 3 phases moteur et Résistance de freinage externe
X11	RESOLVEUR	Entrée retour position moteur (si résolveur)
X12	ANALOG I/O	Entrées et sorties analogiques
X13	SINCOS	Entrée retour position moteur (si codeur SINCOS)

Attention au câblage du connecteur X10. Une mauvaise connexion peut endommager gravement le variateur. X10 comporte également des tensions dangereuses qui peuvent atteindre 900V.

Attendre 5mn après coupure de l'alimentation réseau, avant de déconnecter X10.

2-5- Montage

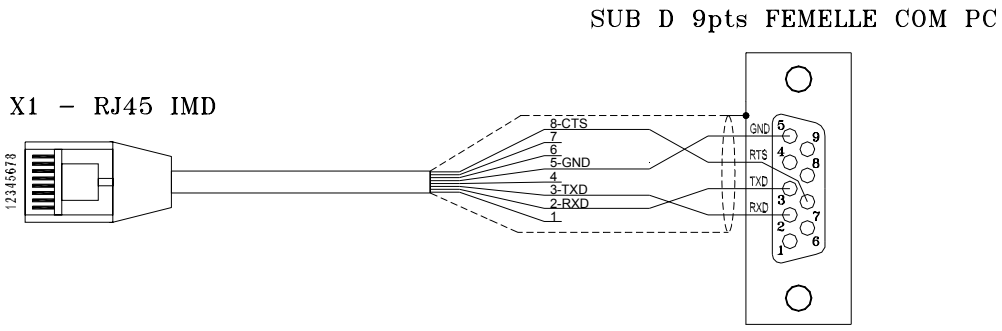
On peut installer plusieurs variateurs les uns à côté des autres en respectant les espaces de séparation pour une bonne convection naturelle (laisser un espace minimum de 20 mm entre deux variateurs). Laisser un espace supérieur à 90 mm au dessus et dessous des variateurs pour le passage des câbles et la mise en place des connecteurs.


90
55
293.5
233.5
55
90
X5 - ENCODER INPUT
X4 - ENCODER OUTPUT
PE L1 L2 L3
X8- POWER SUPPLY
3 X 230.....480VAC
For 1 x 230Vac connection, use L1 and L2
Pour une alimentation 230Vac mono, utiliser L1 et L2
X7 - DIGITAL I/O
X6 - AUX. SUPPLY
24V
Intelligent Motion Drive
SERAD 271, route des crêtes 44440 TEILLE
www.serad.fr
Capacitors discharge time : 5 minutes
High voltage on X8-X10 connectors
All the PE pins must be connected
Read manual before installing
Temps de décharge des condensateurs : 5 minutes
Haute tension sur les connecteurs X8-X10
Toutes les bornes PE doivent être raccordées
Lire le manuel avant l'utilisation
X10 - BALLAST / MOTOR
RI RB DC+ DC- PE W V U
X11 RESOLVER
X12 ANALOG -I/O
RACCORDEMENT BLINDAGE MOTEUR

2-6- Affectation et brochages des connecteurs

X1: Port de communication RJ45 pour paramétrage PC

N°	Nom	Type	Description
1			
2	RXD	Inp	Réception des données
3	TXD	Out	Transmission des données
4			
5	GND		0V
6			
7			
8	CTS	Inp	Activation liaison système
⚠	SHIELD		Raccordement de la tresse blindée sur le corps du SUBD



X2 et X3: Extension: Bus de communication optionnel RJ45

N°	Module RS 232	Module RS 422	Module RS 485	Module CANopen
1				
2	RXD	RX+		
3	TXD	RX-		
4				
5	GND	GND	GND	GND
6				
7		TX-	TRX-	CAN_L
8		TX+	TRX+	CAN_H
⚠	**SHIELD - Raccorder la tresse blindée sur le corps du SUBD**			

- Les deux connecteurs X2 et X3 sont identiques et contiennent les mêmes signaux. Ils facilitent la mise en réseau de plusieurs variateurs
- Numéro d'adresse (NodeID): Pour les modules RS422, RS485 et CANopen, le NodeID correspond à la valeur des 5 premiers dipswitchs + 1

  Ex: dipswitchs: 1 -> ON, 2 -> OFF, 3 -> ON, 4 -> OFF, 5 -> OFF

  Valeur dipswitchs = 1 + 4 = 5

  NodeID = 5 + 1 = 6
- La validation des résistances de terminaison du bus (120Ω) se fait en activant le dipswitch 6 sur la position ON.

⚠ **En RS232, 1 seul connecteur doit être relié, la communication en RS232 n'autorisant le dialogue qu'entre 2 périphériques (ex : 1 PLC et 1 drive iMD).**

X4: Sortie multifonctions :

- Sortie émulation codeur
- Sortie IMDbus

Une seule fonction est disponible à la fois. Le choix se fait à partir du logiciel iDPL

Connecteur SUBD 9 points femelle

N°	Nom	Type	Emulateur codeur	IMDbus
1	A	Out	Voie A	Data
2	/A	Out	Voie A complémentée	/Data
3	B	Out	Voie B	Clock
4	/B	Out	Voie B complémentée	/Clock
5	Z	Out	Voie Z	NC
6	/Z	Out	Voie Z complémentée	NC
7				
8	GND		0V	0V
9				
⚠	**SHIELD**		**Raccordement de la tresse blindée sur le corps du SUBD**	

NC (non connecté): il est impératif de ne rien raccorder sur ces bornes.

X5: Entrée codeur multifonctions :

- Entrée codeur incrémental
- Entrée codeur absolu SSI
- Entrée stepper
- Entrée IMDbus

Codeur 5V TTL (0-5V, différentiel)

Une seule fonction est disponible à la fois. Le choix se fait à partir du logiciel iDPL.

Connecteur SUBD 9 points mâle

N°	Nom	Type	Codeur incremental	Codeur SSI	Stepper	IMDbus
1	A	Inp	Voie A	Data	Direction	Data
2	/A	Inp	complémentée	/Data	/Direction	/Data
3	B	Inp	Voie B	NC	Pulse	Clock
4	/B	Inp	complémentée	NC	/Pulse	/Clock
5	Z	I/O	Voie Z	Clock	NC	NC
6	/Z	I/O	complémentée	/Clock	NC	NC
7	+5Vdc	Out	externe 100 mA maxi *	NC	NC	NC
8	GND		0V	0V	0V	0V
9		Inp	NC	Sélection SSI : Relier les pins 8 et 9	NC	NC
	SHIELD		Raccordement de la tresse blindée sur le corps du SUBD			

* Si le retour position est de type SINCOS, alors ne pas utiliser l'alimentation 5V (pin7 du connecteur X5) mais une source d'alimentation externe.

NC (non connecté): il est impératif de ne rien raccorder sur ces bornes.

X6: Alimentation auxiliaire 24 Vdc

Connecteur débrochable 2 points au pas de 5,08 mm

N°	Nom	Type	Description
1	XGND		0V
2	24Vdc	Inp	Alimentation carte, backup position moteur

X7: Entrées / sorties logiques

Connecteur débrochable 8 points au pas de 3,81 mm

N°	Nom	Type	Description
1	Q2	Out	Sortie 2 programmable : **type NPN** * statique 24 Vdc 100mA
2	Q1	Out	Sortie 1 programmable : fonction DRIVE READY en standard
3	Q1		Type relais contact NO entre les bornes 2 et 3
4	DGND		0V entrées / sorties logiques
5	I4	Inp	Entrée 4 programmable rapide
6	I3	Inp	Entrée 3 programmable rapide
7	I2	Inp	Entrée 2 programmable
8	I1	Inp	Entrée 1 programmable: fonction ENABLE en standard

La sortie Q2* type collecteur ouvert : retour de 0V ⇒ la charge doit être branchée entre Q2 et le +24Vcc.

X8: Alimentation réseau,

Connecteur débrochable 4 points au pas de 7,62 mm

N°	Nom	Type	Description
1	PE		Terre réseau
2	L1	Inp	Phase L1 pour réseau 230V et réseau 400V
3	L2	Inp	Neutre pour réseau 230V ou phase L2 pour réseau 400V
4	L3	Inp	Phase L3 réseau 400V

Attention au câblage du connecteur X10. Une mauvaise connexion peut endommager gravement le variateur. X10 comporte également des tensions dangereuses.

Le câble moteur blindé doit arriver directement sur les bornes du variateur.

Relier la tresse de blindage sur la fixation prévue à cet effet (voir 2-2 Vue de face).

X9: Option : Extension 12 entrées / 8 sorties logiques

Connecteur SUBD 25 points femelle

N°	Nom	Type	Description
1	I5	Inp	Entrée 5 programmable
2	I6	Inp	Entrée 6 programmable
3	I7	Inp	Entrée 7 programmable
4	I8	Inp	Entrée 8 programmable
5	I9	Inp	Entrée 9 programmable
6	I10	Inp	Entrée 10 programmable
7	IOGND*		0V entrées / sorties logiques
8	Q3	Out	Sortie 3 programmable
9	Q4	Out	Sortie 4 programmable
10	Q5	Out	Sortie 5 programmable
11	Q6	Out	Sortie 6 programmable
12	IO 24Vdc**	Inp	Alimentation externe 24 Vdc
13	IO 24Vdc**	Inp	Alimentation externe 24 Vdc
14	I11	Inp	Entrée 11 programmable
15	I12	Inp	Entrée 12 programmable
16	I13	Inp	Entrée 13 programmable
17	I14	Inp	Entrée 14 programmable
18	I15	Inp	Entrée 15 programmable rapide
19	I16	Inp	Entrée 16 programmable rapide
20	Q7	Out	Sortie 7 programmable
21	Q8	Out	Sortie 8 programmable
22	Q9	Out	Sortie 9 programmable
23	Q10	Out	Sortie 10 programmable
24	IOGND*		0V entrées / sorties logiques
25	IOGND*		0V entrées / sorties logiques
	SHIELD		Raccordement de la tresse blindée sur le corps du SUBD

*Pins 7, 24 et 25 : connexion interne **Pins 12, 13 : connexion interne

X10: Alimentation moteur et résistance de freinage externe

Connecteur débrochable 8 points au pas de 7,62 mm

N°	Nom	Type	Description
1	RI		Résistance de freinage interne *
2	RB		Résistance de freinage *
3	DC Bus +	Out	Bus continu
4	DC Bus -	Out	Bus continu
5	PE		Terre moteur
6	W	Out	Phase W moteur
7	V	Out	Phase V moteur
8	U	Out	Phase U moteur

Le câble moteur blindé doit arriver directement sur les bornes du variateur.

Relier la tresse de blindage sur la fixation prévue à cet effet (voir 2-2 Vue de face).

*Sélection de la résistance de freinage :

- Résistance interne : Mettre un shunt entre les bornes 1 et 2

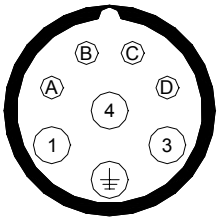
- Résistance externe : Enlever le shunt entre les bornes 1 et 2

Raccorder la résistance externe entre les bornes 2 et 3

La tension sur le connecteur X10 peut atteindre 900V!

Attendre 5mn après coupure de l'alimentation réseau, avant de déconnecter X10.

MOTEUR SERAD

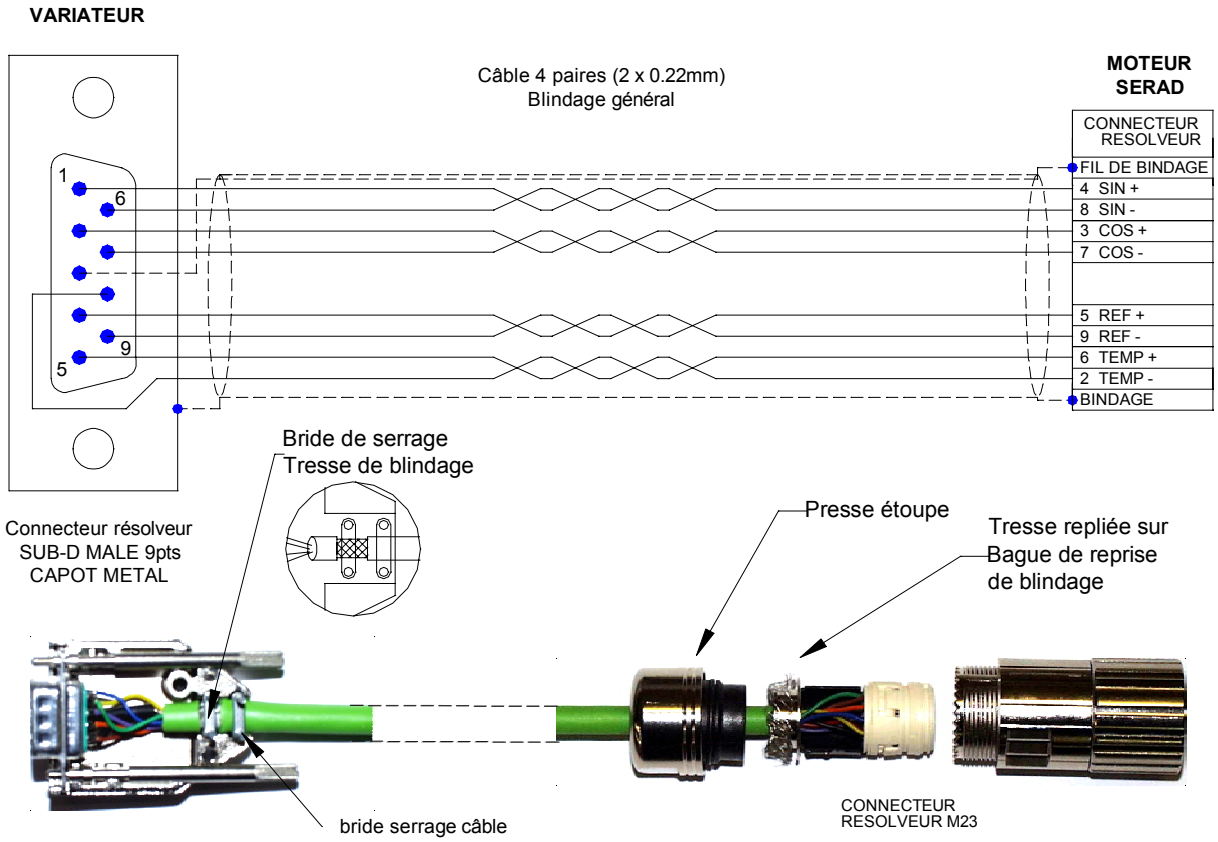


Brochage	
1	Phase U
4	Phase V
3	Phase W
2	Terre
C	Frein +
D	Frein -

X11: Entrée retour position moteur (résolveur)

Connecteur SUBD 9 points femelle

N°	Nom	Type	Description
1	S2	Inp	Voie sinus
2	S1	Inp	Voie cosinus
3	AGND		0V analogique
4	R1	Out	Excitation
5	°CM+	Inp	Capteur température moteur
6	S4	Inp	Référence voie sinus
7	S3	Inp	Référence voie cosinus
8	°CM-	Inp	Référence capteur température moteur
9	R2	Out	Référence excitation
⚠	**SHIELD**		**Raccordement de la tresse blindée sur le corps du SUBD**



X12: Entrées / sorties analogiques

Connecteur SUBD 9 points mâle

N°	Nom	Type	Description
1	IN2 -	Inp	Entrée analogique 2
2	IN2+	Inp	Entrée analogique 2 : consigne limitation de couple
3	IN1-	Inp	Entrée analogique 1
4	IN1+	Inp	Entrée analogique 1 : consigne vitesse ou couple suivant le mode
5	AGND		0V analogique
6	-12V	Out	Sortie -12V, 20 mA
7	AGND		
8	+12V	Out	Sortie +12V, 20 mA
9	OUT	Out	Sortie analogique fonction monitoring
	SHIELD		**Raccordement de la tresse blindée sur le corps du SUBD**

X13: Option : Entrée codeur SinCos

Connecteur SUBD 15 points mâle

N°	Nom	Type	Description
1	°CM +	Inp	Capteur température moteur
2	AGND		0V analogique
3	/DATA	I/O	/DATA (EnDat*) /RS485 (HIPERFACE)
4	/CLK	Out	/CLOCK (EndDat*)
5	+5V	Out	Sortie +5V, 200 mA (EnDat*)
6			
7	REFCOS	Inp	Référence voie cosinus
8	REFSIN	Inp	Référence voie sinus
9	°CM-	Inp	Référence capteur température moteur
10	+8,3V	Out	Sortie +8.3V, 150 mA (HIPERFACE)
11	DATA	I/O	DATA (EnDat*) RS485 (HIPERFACE)
12	CLK	Out	CLOCK (EndDat*)
13			
14	COS	Inp	Voie cosinus
15	SIN	Inp	Voie sinus
	SHIELD		**Raccordement de la tresse blindée sur le corps du SUBD**

* EnDat en cours de développement

2-7- Câbles

Nous vous proposons tous les câbles avec les connecteurs montés. Ils sont disponibles en différentes qualités (standard, robotique pour les chaînes porte-câble, etc.), nous consulter.

- **Câble COM de communication RS 232 X1 :**

  Câble blindé, 4 fils

  Tresse de blindage relié à chaque extrémité au capot des SUBD et RJ45.

- **Câble ENCODER X4/X5 :**

  Câble avec blindage général, 4 paires torsadées 0.25 mm².

  Tresse de blindage relié à chaque extrémité au capot des SUBD.

- **Câble ANALOG X12 :**

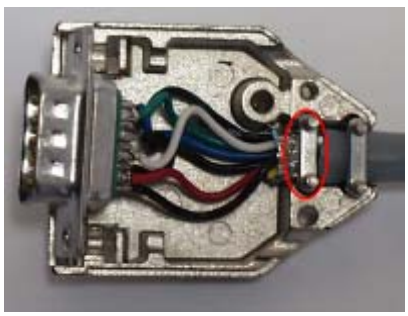
  Câble blindé 2 fils 0.25 mm² par entrée analogique.

  Tresse de blindage à relier côté variateur sur la vis prévue à cet effet (voir vue de face avant du variateur) et relier l'autre extrémité au châssis de l'appareil (exemple : commande d'axes …).

- **Câble FEEDBACK retour moteur (resolver) X11 :**

  Câble avec blindage général, 4 paires torsadées 0.25 mm².

  Raccordement de la tresse de masse au SUBD résolveur comme sur la photo ci-dessous :



- **Câble POWER moteur X10 :**

  Câble avec blindage général 4 fils (plus 2 fils si frein).

  Section 1,5 mm² pour variateur jusqu'à 8A. Au delà, prévoir du 2,5 mm².

  La tresse de blindage est à relier côté variateur sur la vis prévue à cet effet (voir vue de face avant du variateur).

2-8- Schémas de raccordement / Protection :

Toutes les connexions doivent être réalisées par des personnes qualifiées. Les câbles doivent être testés avant d'être connectés, toute mauvaise connexion peut entraîner de graves dysfonctionnements.

Mettre hors tension le variateur avant d'insérer ou de retirer des connecteurs.

S'assurer que la borne de terre du connecteur de l'alimentation du variateur est bien connectée (borne 4 du connecteur X8).

Connecter la terre du moteur au point de terre du variateur (borne 5 du connecteur X10) avant toute mise sous tension.

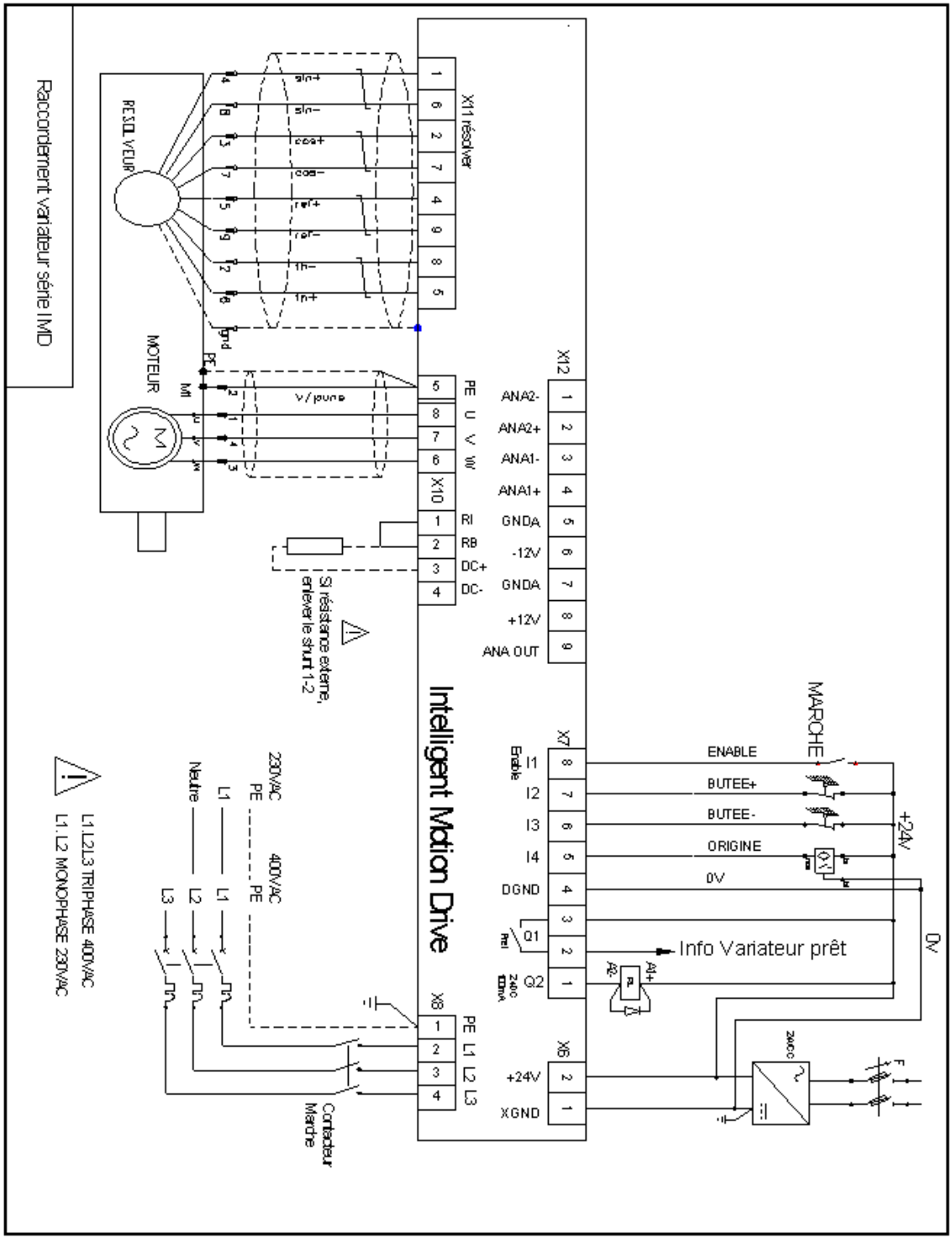
Pour les câbles blindés, raccorder la tresse au châssis à chaque extrémité via les capots des connecteurs (pour les SUBD) ou les vis prévus à cet effet (connecteur X7) afin d'assurer une équipotentialité optimale.

Toute bobine (frein) alimentée par courant continu (24V) doit être obligatoirement pourvue d'une diode de roue libre (ex : 1N4007) afin d'empêcher des surtensions (plus de 80V) qui risqueraient de détériorer l'ensemble de l'électronique.

Drive	Tension d'entrée	Courant d'entrée max	Protection : Disjoncteur courbe C	Section câble
IMD / 1	400V triphasé	2,2A	10A maxi	1,5²
	230V monophasé	3,5A	10A maxi	1,5²
IMD / 2	400V triphasé	4,2A	10A maxi	1,5²
	230V monophasé	7A	10A maxi	1,5²
IMD / 5	400V triphasé	8,2A	10A maxi	1,5²
	230V monophasé	14A	16A maxi	2,5²
IMD / 10	400V triphasé	16,2A	20A maxi	2,5²
IMD / 20	400V triphasé	32,2A	32A maxi	4²

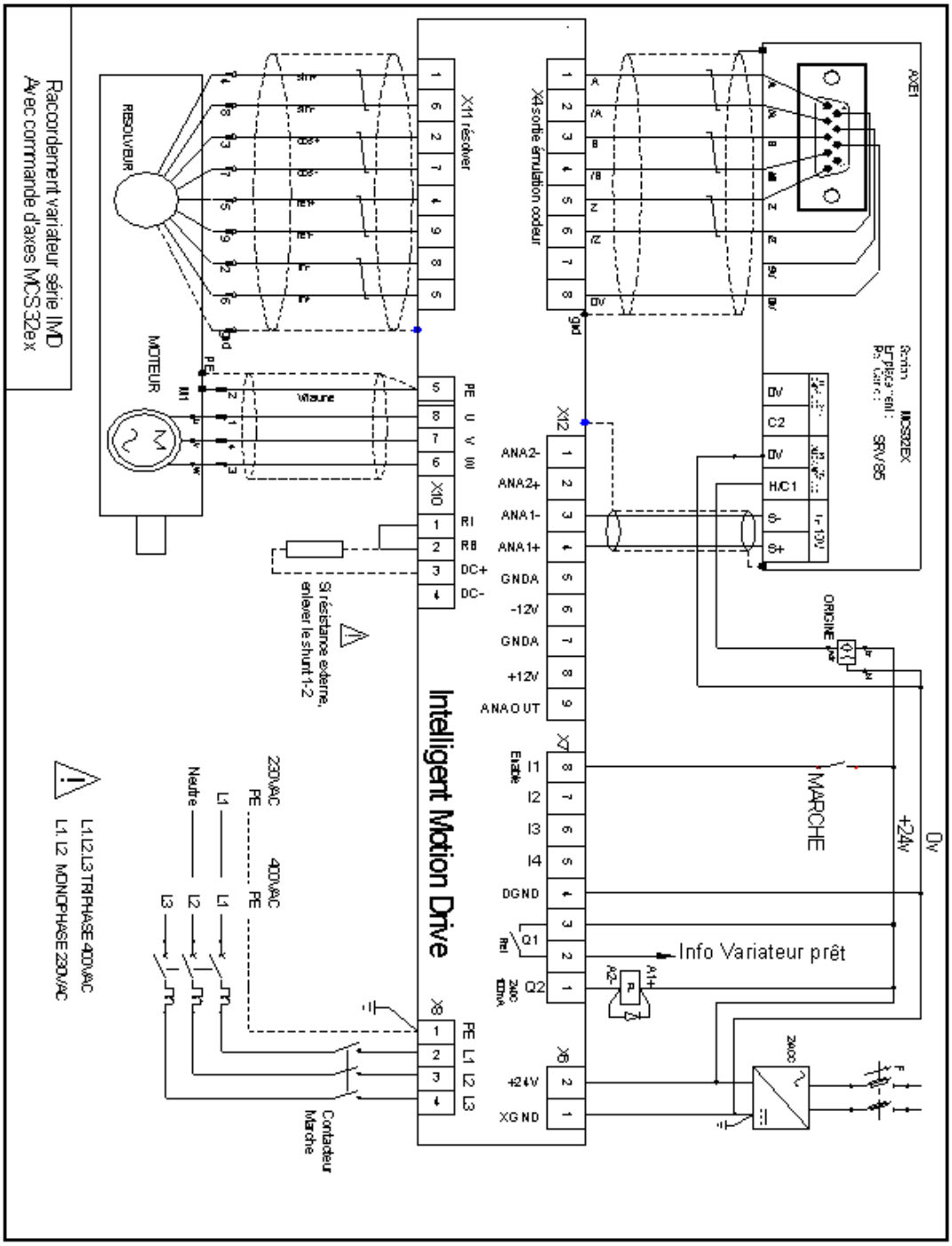
Attention : Le courant d'appel pour chaque variateur est de 25A pendant 10ms.

2-9- Variateur autonome



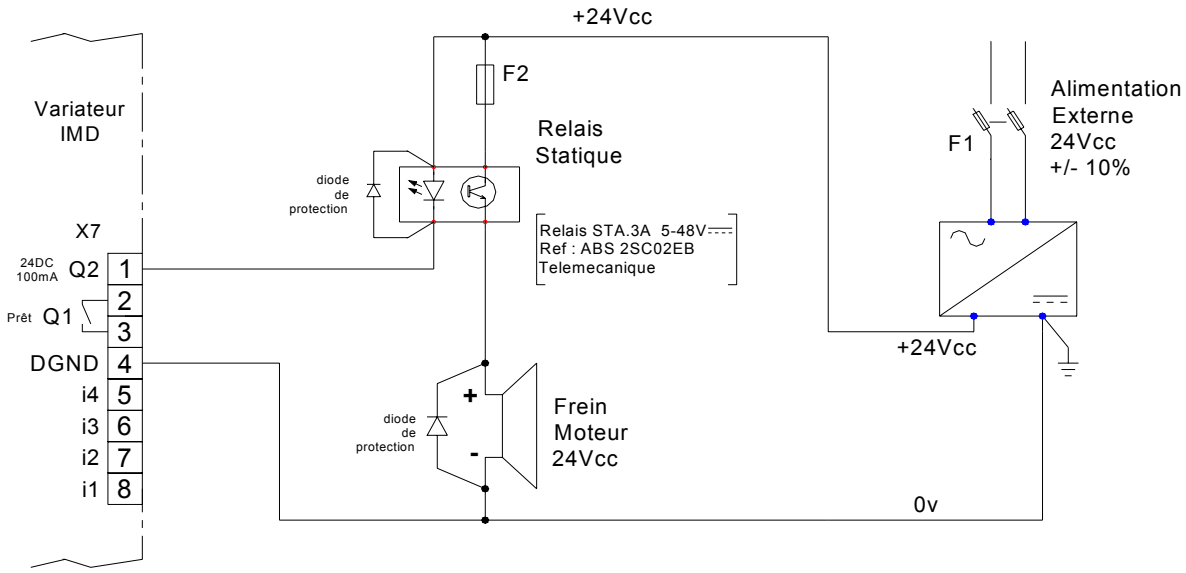
La sortie Q2 est du type NPN (collecteur ouvert) 100 mA maxi. La charge doit être branchée entre Q2 et le +.

2-10- Variateur piloté par une commande d'axe



La sortie Q2 est du type NPN (collecteur ouvert) 100 mA maxi. La charge doit être branchée entre Q2 et le +24 Vdc.

2-11- Raccordement d'un frein moteur



La sortie Q2 est du type NPN (collecteur ouvert) 100 mA maxi. La charge doit être branchée entre Q2 et le +24Vdc.

A partir du logiciel iDPL de paramétrage, aller dans le menu Paramètres / Entrées-sorties digitales et sélectionner la fonction Frein dans la sortie n°2

Il est obligatoire de mettre les 2 diodes de protection sous peine d'endommager les composants internes du variateur.

2-12- Vérifications avant mise en route

↳ L'entrée ENABLE étant à 0, mettre sous tension l'alimentation auxiliaire 24 Vdc.

↳ S'assurer que l'afficheur de STATUS s'allume.

↳ Mettre la puissance.

↳ Si l'afficheur de STATUS indique un message d'erreur (se reporter à la liste des erreurs).

3- Logiciel iDPL

3-1- Installation du logiciel iDPL

3-1-1- Configuration du système

- **Configuration minimale :**

⇒ PC Pentium II

⇒ RAM 64 Mo

⇒ Disque dur (35 Mo disponibles)

⇒ Microsoft® Windows™ 98 SE, NT, 2000 et XP

⇒ Lecteur de CD-ROM (2X)

⇒ Ecran SVGA

⇒ Souris ou autre périphérique de pointage

- **Configuration recommandée :**

⇒ PC Pentium® III

⇒ RAM 256 Mo

⇒ Disque dur (35 Mo disponibles)

⇒ Microsoft® Windows™ 2000 ou XP

⇒ Lecteur de CD-ROM (4X)

⇒ Ecran SVGA

⇒ Souris ou autre périphérique de pointage

Ce logiciel peut aussi fonctionner sous Microsoft® Windows NT™. Cette application ne travaille pas sous Unix, Mac, MS-DOS et Microsoft® Windows 3.11.

3-1-2- Procédure d'installation du logiciel iDPL

Le logiciel Drive Programming Language est fourni sous forme de CD-ROM.
L'installation du logiciel se fait comme suit:

- Vérifier la configuration requise pour installer le logiciel
- Insérer le CD-ROM dans le lecteur approprié.
- Dans le menu déroulant

, sélectionner

.
- Dans la boîte de dialogue « Exécuter », sélectionner

.
- Dans la boîte de dialogue « Parcourir », sélectionner le lecteur où se situe le CD-ROM.
- Sélectionner

puis

dans la boîte de dialogue « Parcourir ».
- Sélectionner

dans la boîte de dialogue « Exécuter ».

  Le programme d'installation du logiciel iDPL débute.

- Le début de l'installation est marqué par une série de boîte de dialogue guidant l'utilisateur :

  1. répertoire de destination
  2. type d'installation (Typique, compacte ou personnalisée)
  3. sélection du dossier programme

Attention : seul un niveau de répertoire peut-être créé.

L'installation des fichiers débute et est indiquée par l'évolution d'un barre graphe.

L'installation se termine par l'ajout de l'icône du logiciel iDPL dans le dossier programme.

3-1-3- Les répertoires

Le logiciel iDPL est installé par défaut dans le répertoire suivant :

C:\Program Files\SERAD\iDpl\

Il contient 5 sous répertoires :

- Data : contenant les sources du logiciel.
- Help : contenant l'aide du logiciel.
- Lib : contenant les différents fichiers de paramétrage du variateur.
- Os : contenant le système d'exploitation du variateur.
- Doc : contenant les fichiers de documentation (description table modbus/CANopen, fichier EDS ...).

3-2- Présentation

3-2-1- Les modes d'utilisation

Pour communiquer avec le variateur, il vous faut le câble CIMDP, permettant de connecter le PC au variateur.

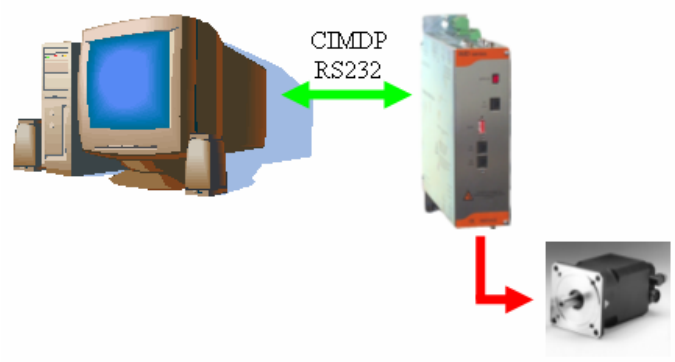
Lors de la connexion PC/variateur, les paramètres sont automatiques reçu dans le logiciel iDPL.

La modification d'un paramètre dans le logiciel iDPL modifie aussi ce paramètre dans le variateur (il n'est pas sauvegardé en cas de coupure).

Le projet par défaut se trouve dans le répertoire Projet du soft iDPL.

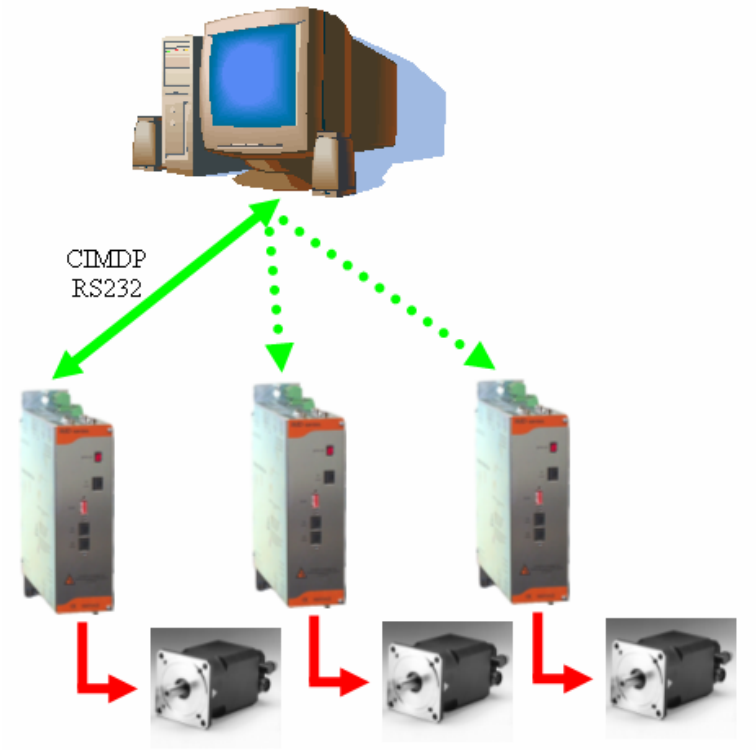
En mode hors-ligne, il est obligatoire d'ouvrir un projet et un fichier de paramètre.

A) Communication avec un variateur seul :



A partir du logiciel, créer un nouveau projet avec un seul variateur.

B) Communication avec plusieurs variateurs :

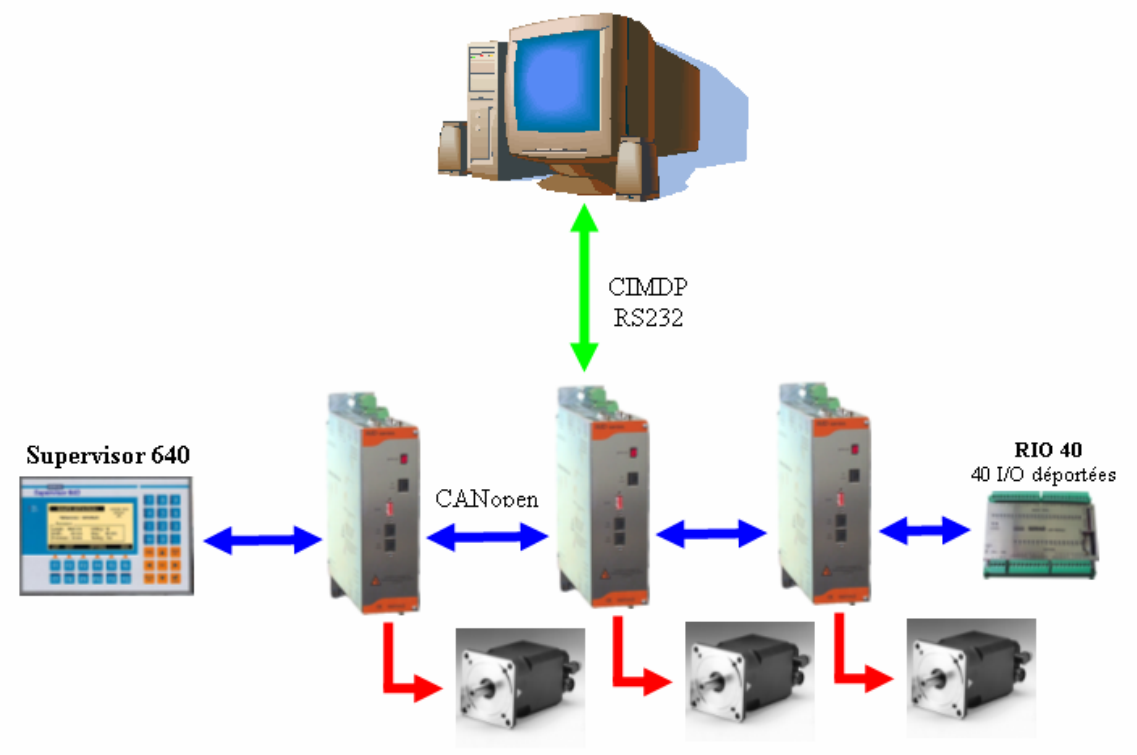


A partir du logiciel, créer un nouveau projet avec le nombre de variateur du système.

Pour changer de variateur, sélectionner un variateur dans la liste de variateurs du logiciel puis brancher le câble CIMDP sur le bon variateur.

C) Communication en Multi drives :

Le multi drive (plusieurs variateurs connectés en réseau CAN) permet le développement sur plusieurs drives simultanément avec seulement un variateur connecté au PC.



A partir du logiciel, créer un nouveau projet avec le nombre de variateur du système.

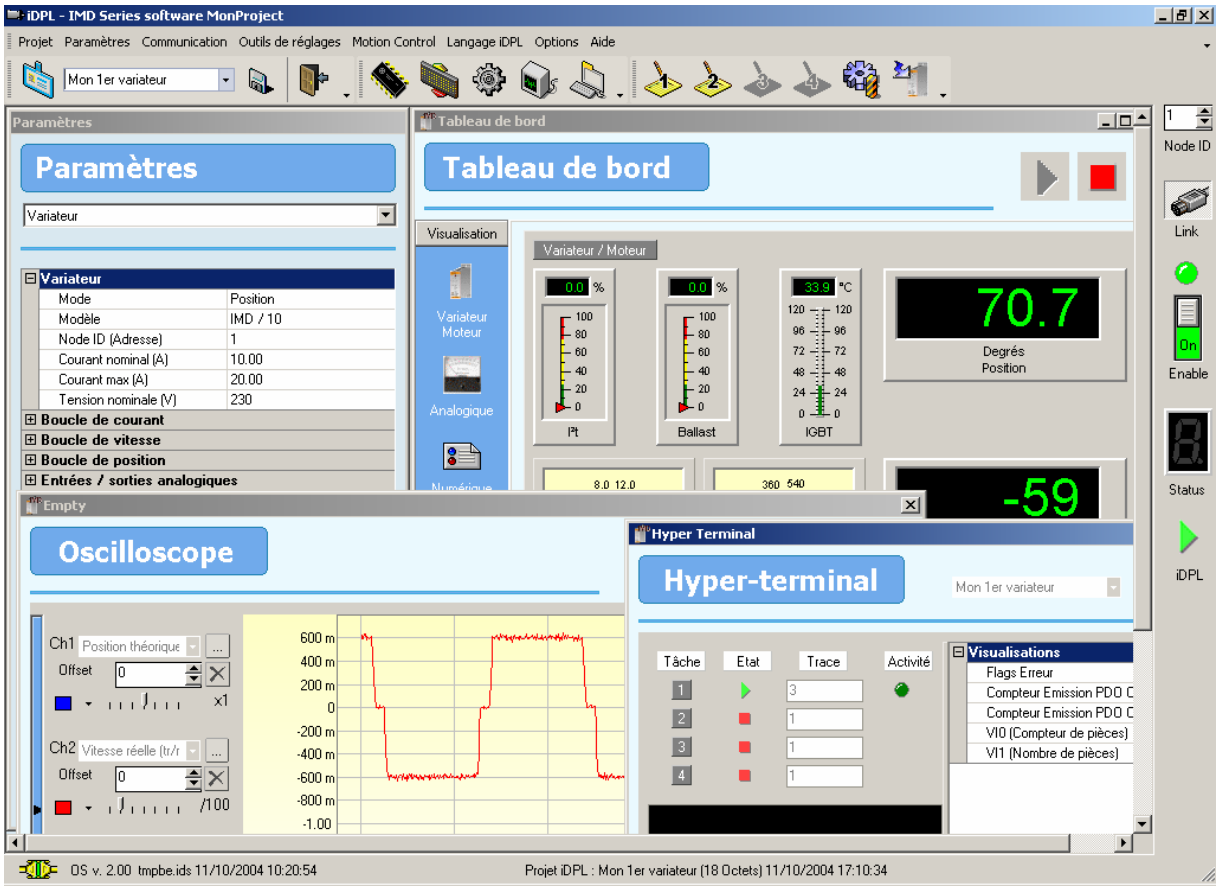
Pour changer de variateur, sélectionner un variateur dans la liste de variateurs du logiciel. Le câble de communication PC/variateur reste fixe sur n'importe quel variateur.

Le PC occupe le noeud n°1 donc l'adressage de votre réseau CAN doit commencer à partir de 2.

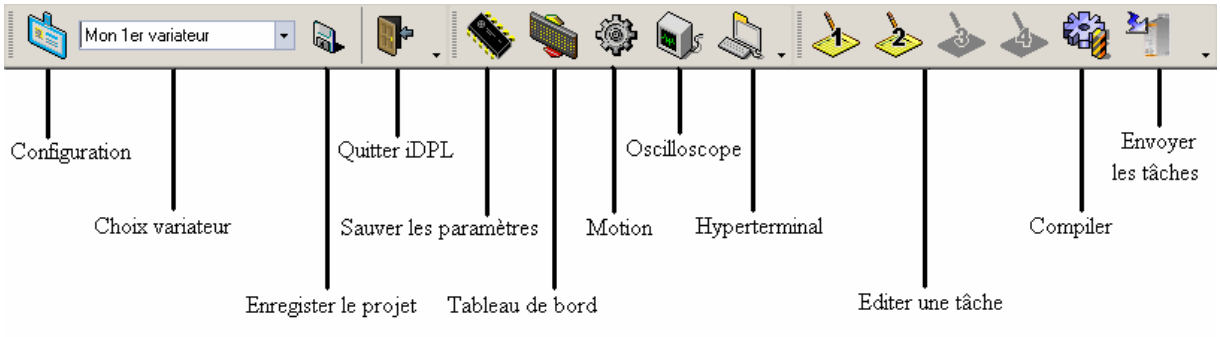
Il est **IMPERATIF** d'être en **communication système** entre le PC et les variateurs pour les projets multi drives.

3-2-2- Ecran initial

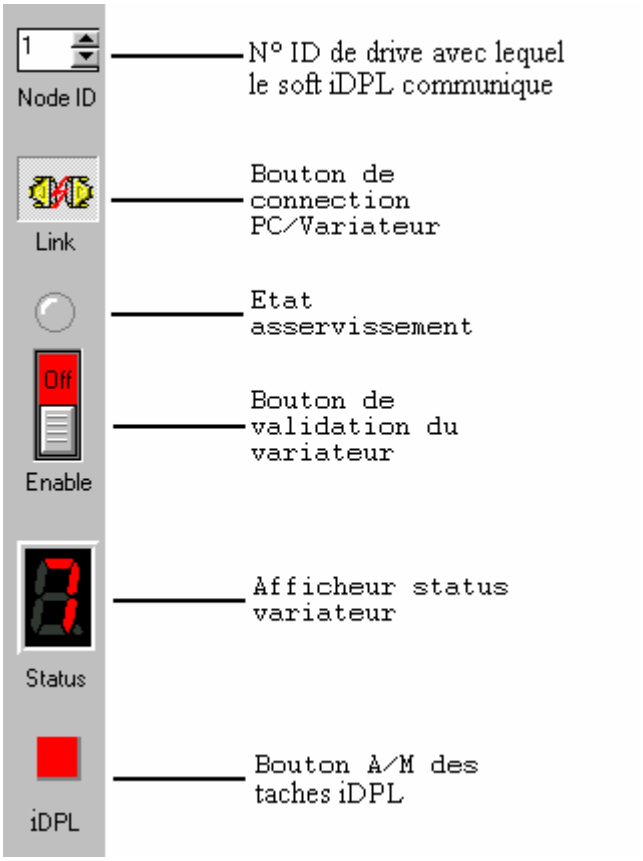
Le logiciel iDPL est caractérisé par une fenêtre principale comportant le menu principal, une barre d'icônes, une barre de commande, une barre d'état et une zone multi-fenêtrage. Les propriétés du multi-fenêtrage permettent à l'utilisateur de pouvoir passer d'une fenêtre à une autre avec un rafraîchissement automatique des informations.



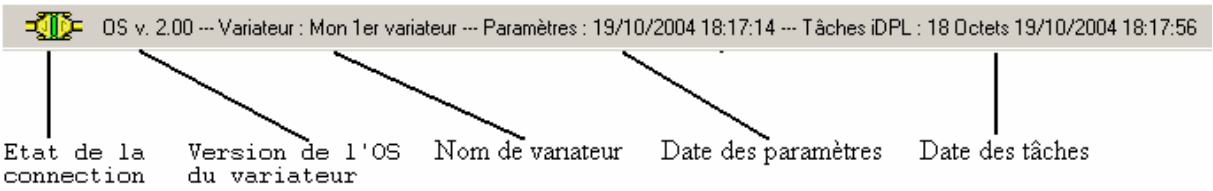
- La barre d'icônes :



- La barre de commande :

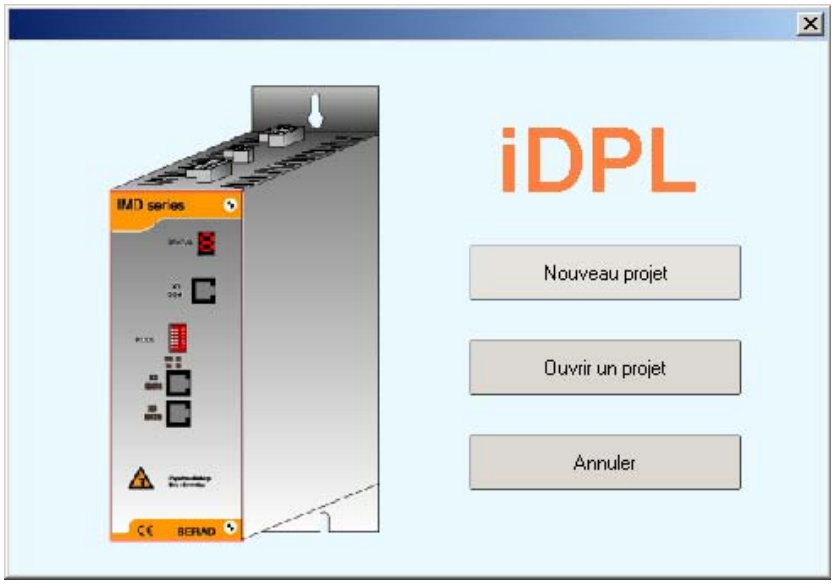


- La barre d'état :


OS v. 2.00 --- Variateur : Mon 1er variateur --- Paramètres : 19/10/2004 18:17:14 --- Tâches iDPL : 18 Octets 19/10/2004 18:17:56
Etat de la connection
Version de l'OS du variateur
Nom de variateur
Date des paramètres
Date des tâches

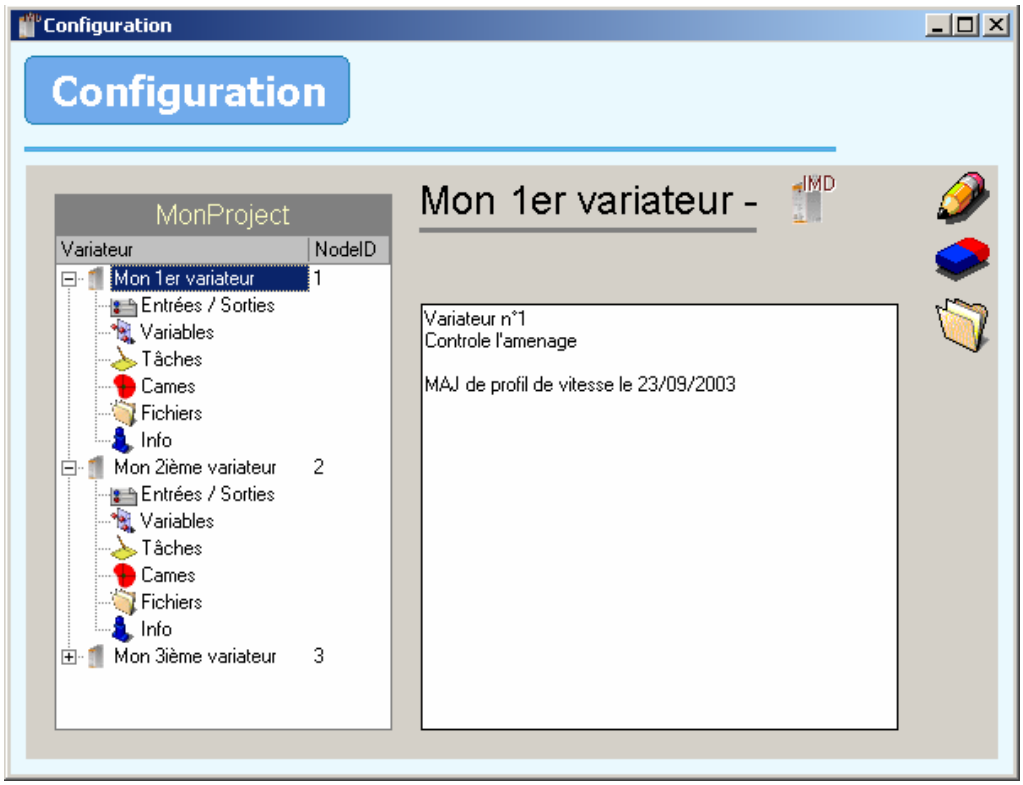
3-2-3- Gestion d'un projet

L'ouverture du logiciel commence par le choix d'un projet :



Il est nécessaire d'ouvrir ou de créer un projet pour accéder au paramétrage du variateur.

Par accéder à la configuration de votre projet, cliquer sur l'icone ou cliquer sur **Configuration** dans le menu **Projet**.



A partir de cette fenêtre vous pouvez déclarer l'ensemble des variateurs du projet ainsi que leurs paramètres, entrées/sorties, variables, tâches, cames ...

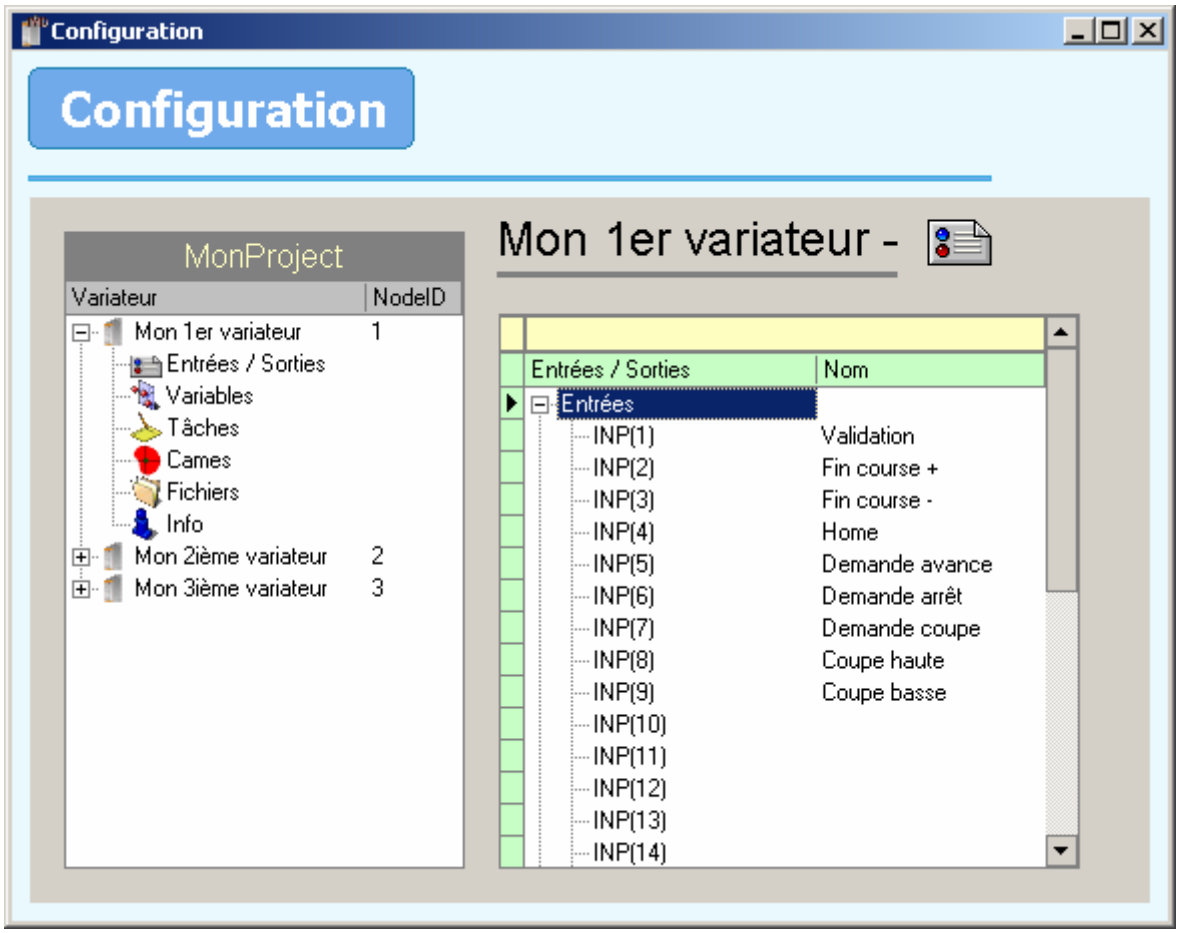
Double cliquer sur le numéro de Node ID pour le modifier et le faire correspondre au à celui des dipswitchs du variateur.

Dans la zone de droite, le programmeur peut laisser des notes pour un meilleur suive du projet.

Un projet peut gérer jusqu'à 127 variateurs.

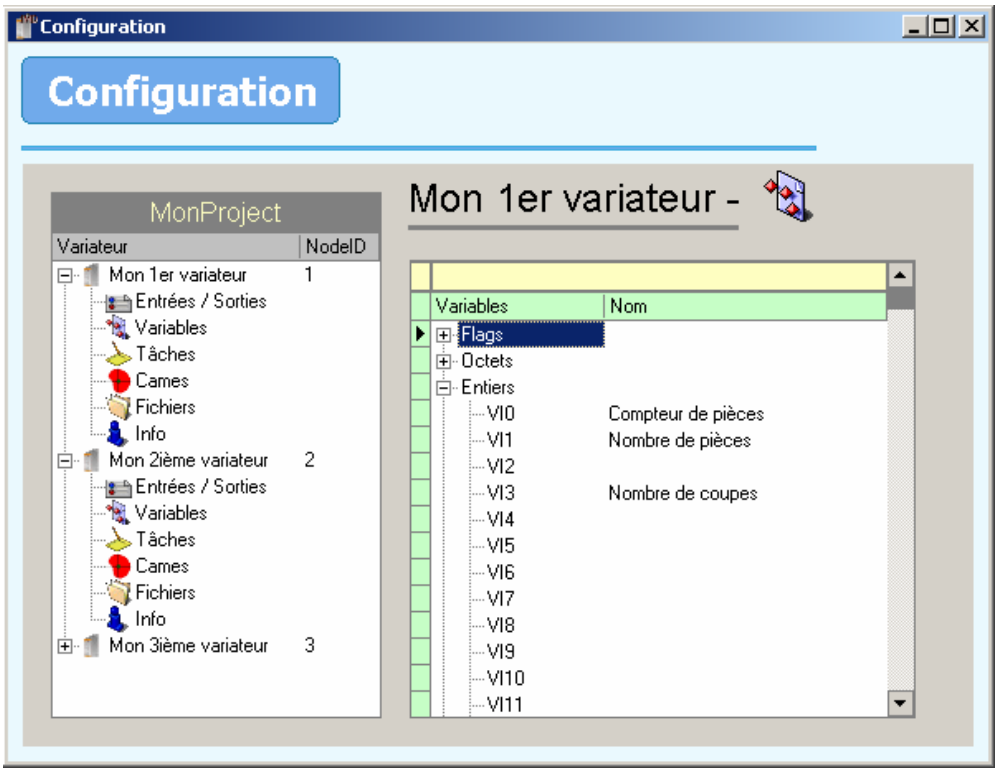
Les icônes de droites permettent d'ajouter, supprimer ou importer un variateur

- Déclaration des noms des entrées/sorties :



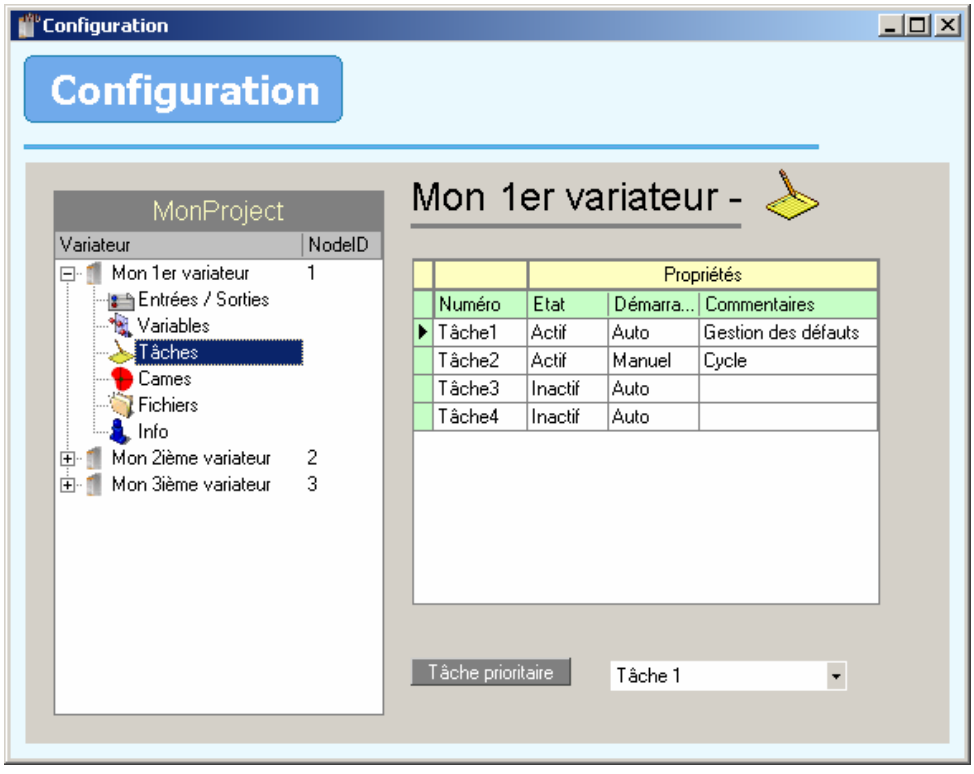
Le nom donné à chaque entrée/sortie peut être utilisé dans les tâches du variateur afin de faciliter la lecture du programme.

➢ Déclaration des noms de variables :



Le nom donné à chaque variable peut être utilisé dans les tâches du variateur afin de faciliter la lecture du programme.

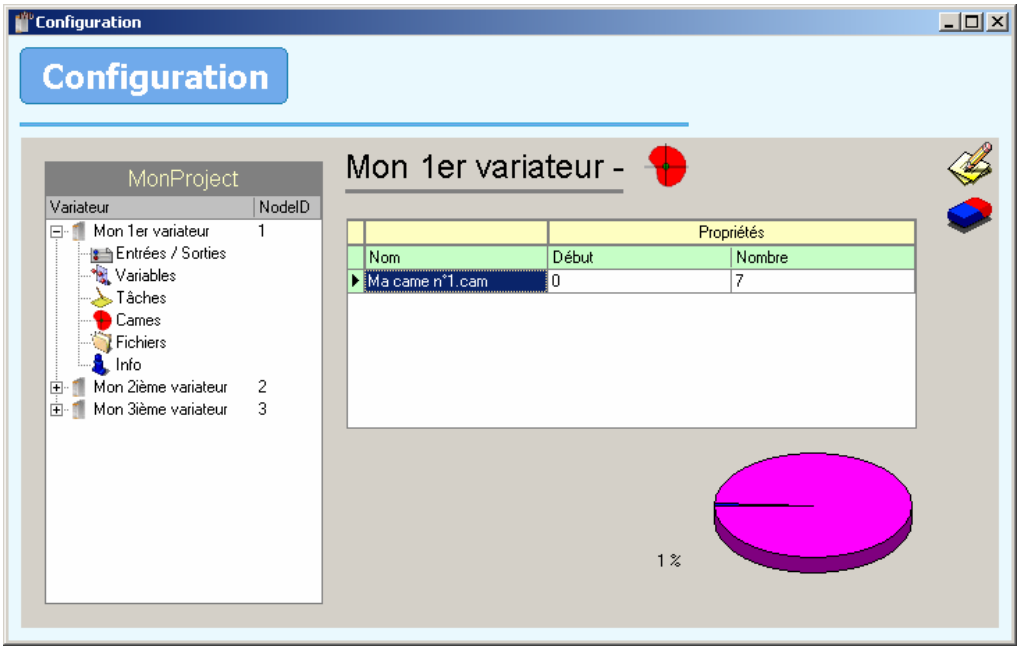
➢ Déclaration des tâches :



Permet d'activer les différentes taches (avec un démarrage à la mise sous tension ou par commande manuelle).

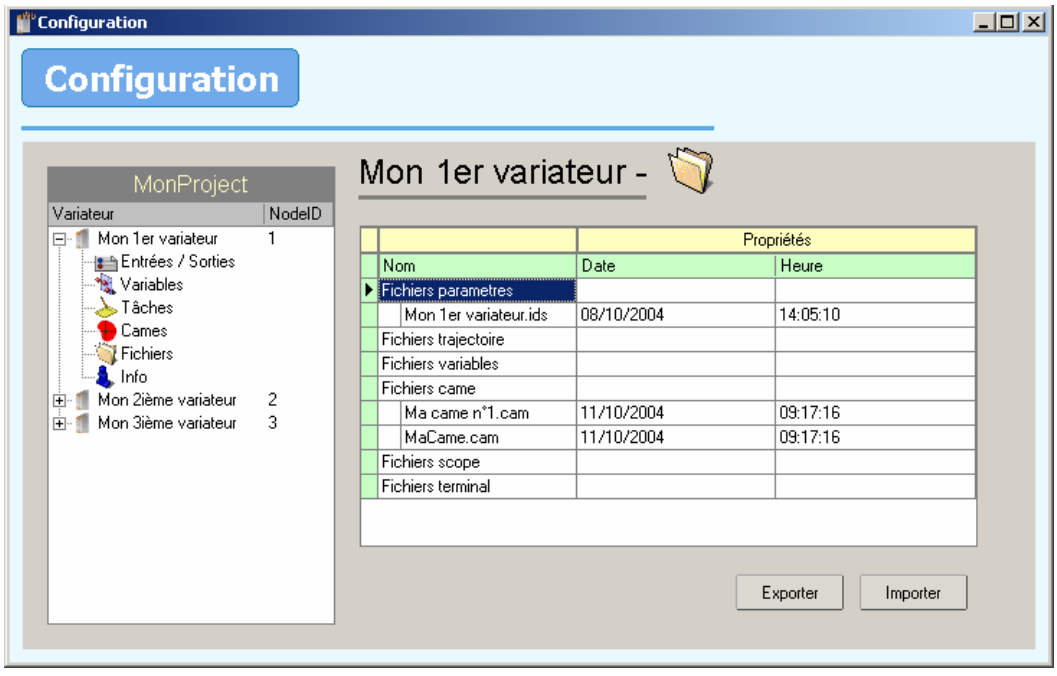
Il est aussi possible de définir la tâche prioritaire (priorité des taches).

- Déclaration des cames :



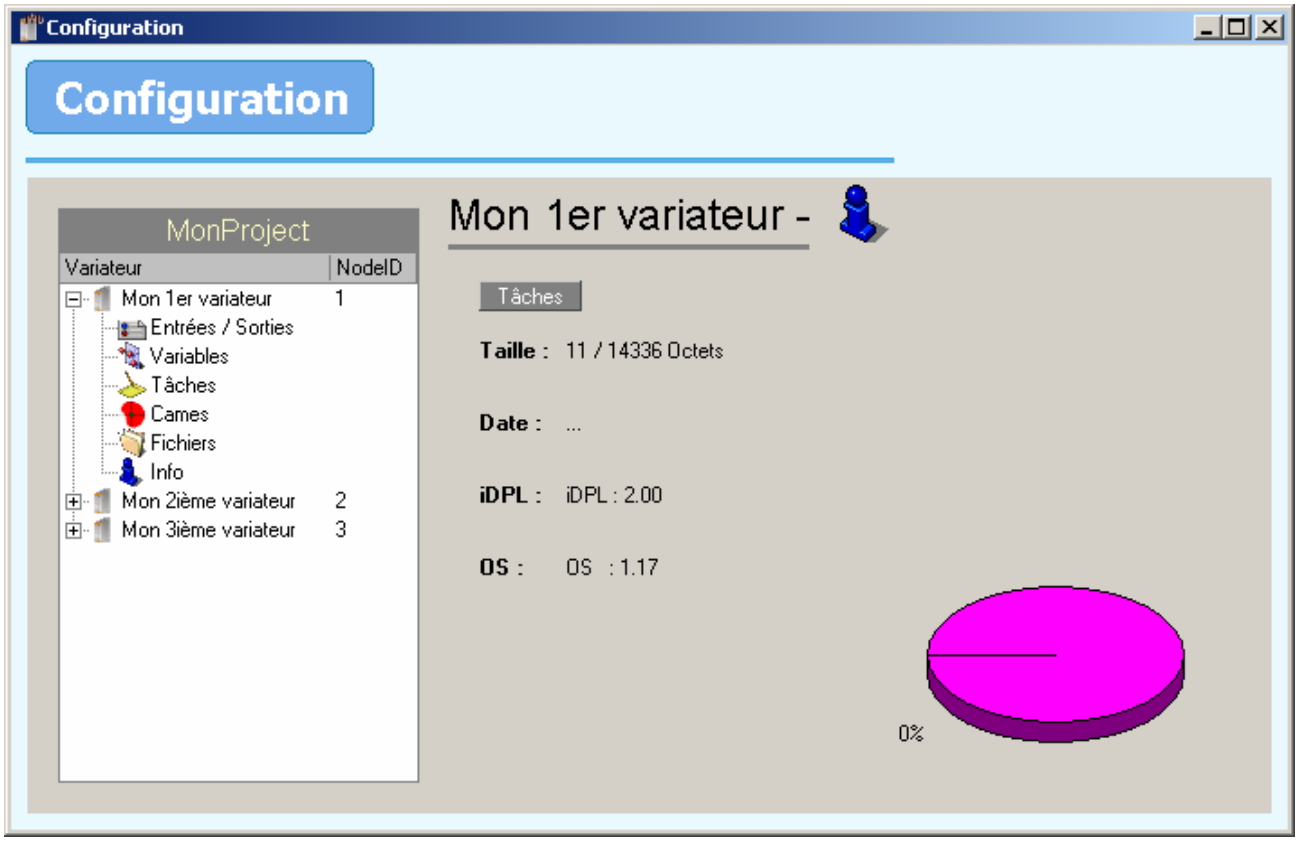
Permet de déclarer les différentes cames selon la taille et leur position dans la mémoire FLASH.

- Liste des fichiers du variateur :



Permet de consulter la liste des différents fichiers composants la configuration du variateurs ainsi que le date de création, permet aussi d'importer ou d'exporter des fichiers d'autres projets.

- Info sur le variateur :



Permet de visualiser la version de iDPL et OS du variateur ainsi que la place occupé par les tâches iDPL dans la mémoire.

3-2-4- Contenu d'un projet

Un projet est composé d'un fichier NomDuProjet.idw et d'un répertoire NomDuProjet.Data.

Dans ce répertoire, on y trouve :

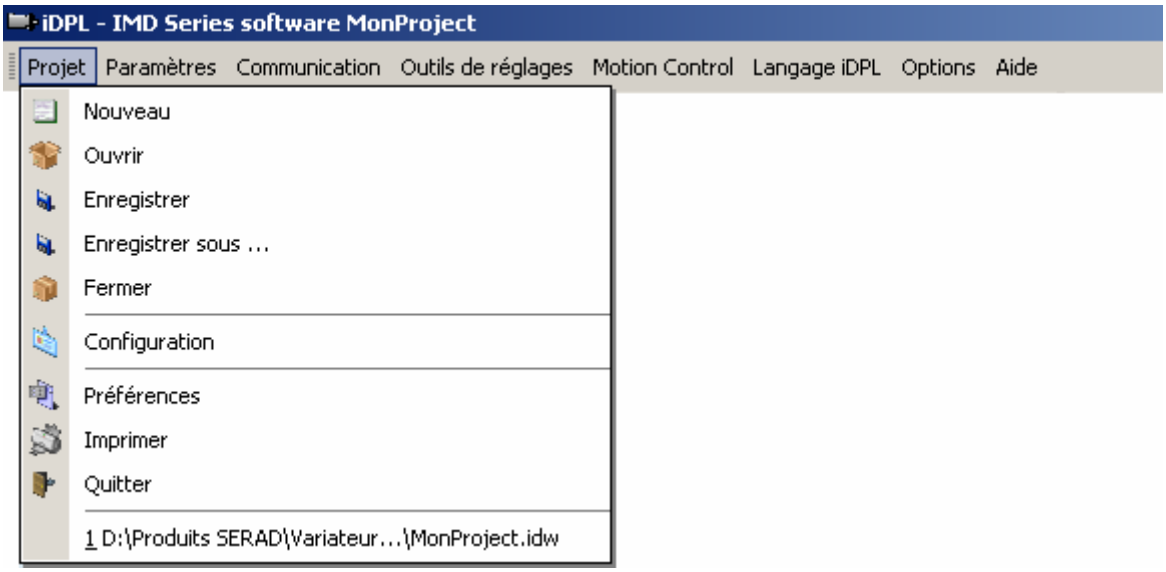
- des fichiers NomDuVariateur.ids contenant les paramètres d'un variateur
- des fichiers NomDuVariateur.idp contenant les informations annexes au variateur :

- les déclarations de entrées/sorties
- les déclarations des variables
- les déclarations des tâches

- des répertoires NomDuVariateur.Data contenant divers fichiers :
  - des fichiers TacheX.dpl contenant les différentes tâches sous format texte.
  - un fichier NomDuVariateur.dpi contenant les informations
  - un fichier NomDuVariateur.dpo contenant la configuration de l'oscilloscope
  - un fichier NomDuVariateur.dpv contenant la liste des variables et leurs valeurs.
  - Un fichier NomDuVariateur.trj contenant les trajectoires du variateur.
  - un répertoire bin contenant les fichiers compilés des tâches et paramétrages nécessaire au variateur.
  - des fichiers .dpt contenant un paramétrage du terminal
  - des fichiers .cam contenant un profil de came

3-3- Menus et icônes

3-3-1- Projet



A) Nouveau :

Icône :

Action : Cette commande permet à l'utilisateur de définir un nouveau projet.

B) Ouvrir :

Icône :

Action : Cette commande ouvre la boîte de dialogue "Ouvrir un Projet". Elle permet à l'utilisateur de spécifier le chemin et le nom du projet à charger.

C) Enregistrer :

Icône :

Action : Cette commande permet la sauvegarde complète du projet en cours sous le nom spécifié.

D) Enregistrer sous :

Icône :

Action : Cette commande ouvre la boîte de dialogue "Enregistrer sous" et permet à l'utilisateur de spécifier le nom du projet de sauvegarde. Cette commande a pour effet de créer un fichier et un répertoire portant le nom spécifié avec pour le premier l'extension "idw" et pour le second l'extension "data".

E) Fermer :

Icône :

Action : Cette commande ferme le projet en cours.

F) Déclarations :

Icône :

Action : Permet de déclarer les tâches, les noms de variables et les noms des E/S.

Voir chapitre sur la gestion d'un projet

G) Préférences :

Icône :

Action : Cette commande permet à l'utilisateur de définir son type d'impression (imprimante, papier, etc…). L'orientation du papier peut-être modifiée mais n'est pas prise en compte lors de l'impression (impression de type portrait).

H) Imprimer :

Icône :

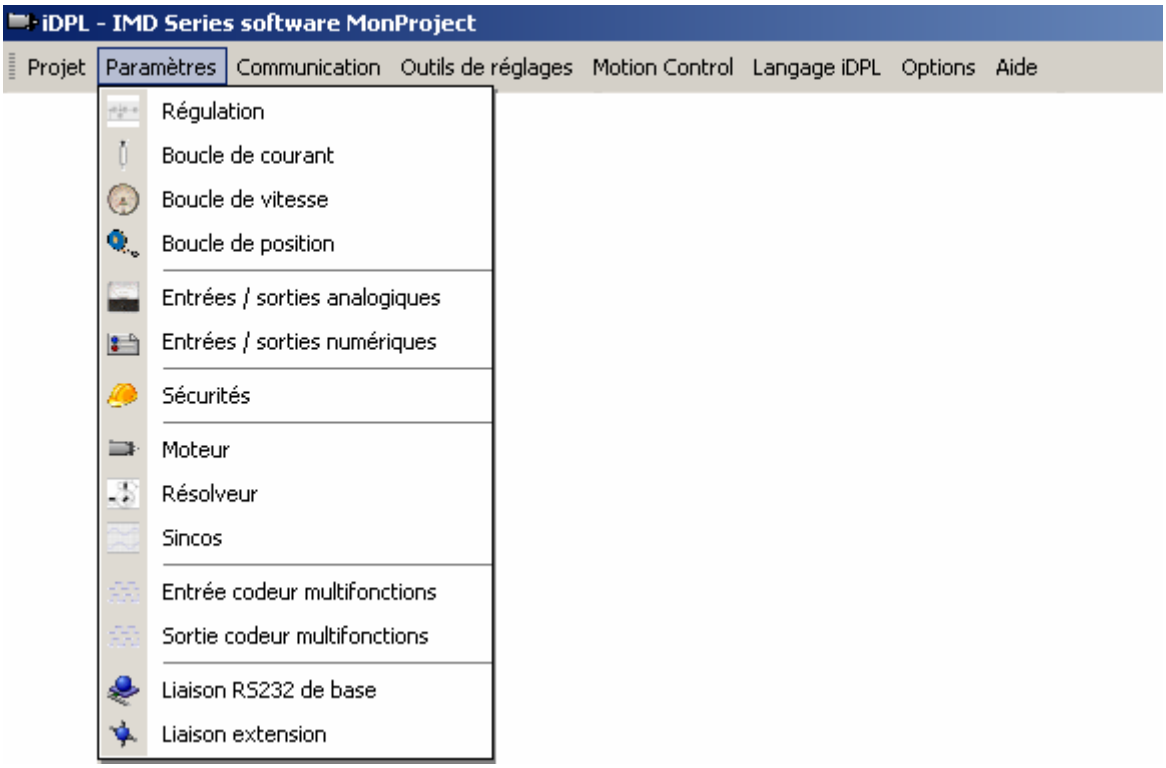
Action : Cette commande réalise une impression totale ou personnalisée du projet.

I) Quitter :

Icône :

Action : Cette commande permet à l'utilisateur de quitter le logiciel.

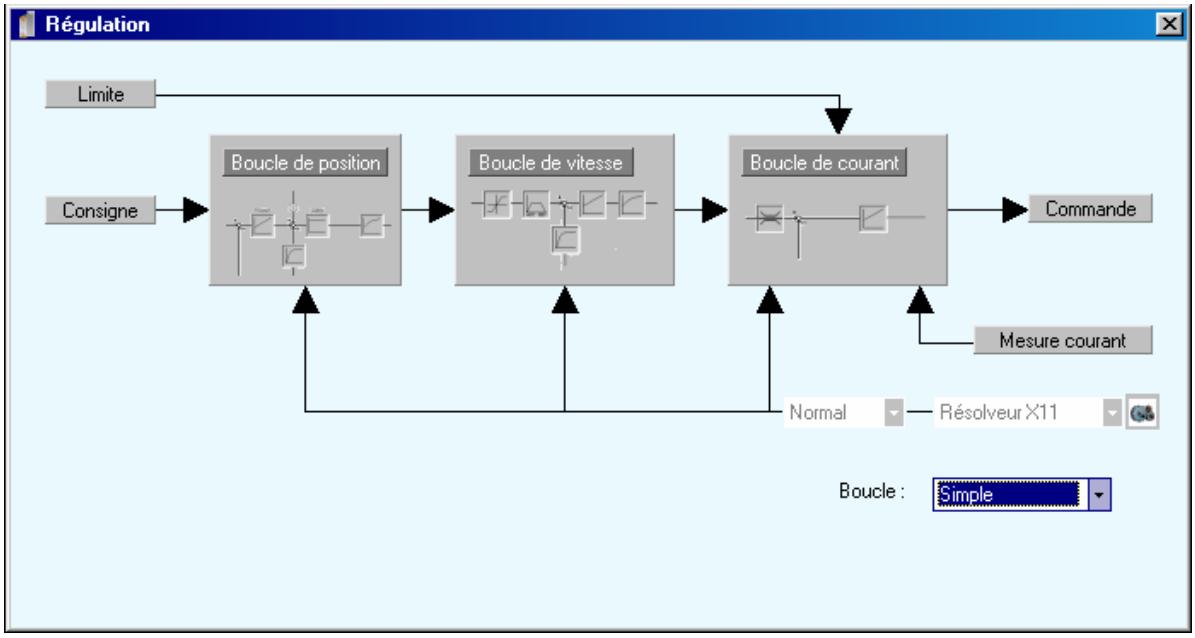
3-3-2- Paramètres



A) Régulation :

Icône :

Action : Fenêtre principale pour le paramétrage de la régulation du variateur. Elle permet d'accéder aux autres fenêtres de régulation et aux fenêtres de configuration.



a) Boucle simple :

Les 3 boucles de régulation utilisent le même retour de position (Résolveur ou SinCos). Il est possible à partir de cet écran de modifier le sens du retour de position.

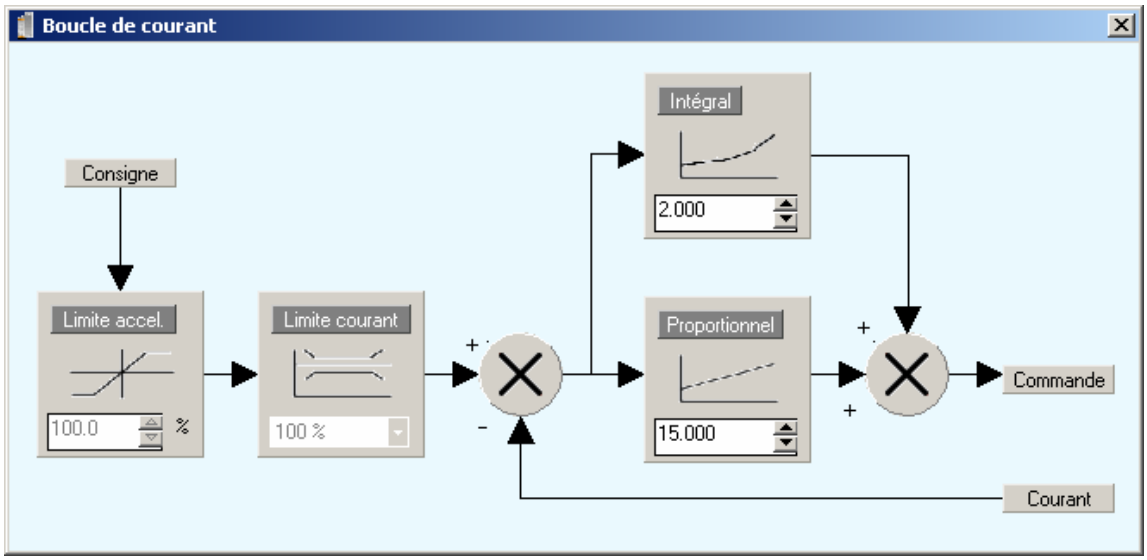
b) Double boucle :

La boucle de position utilise un retour de position (Résolveur ou SinCos) différent des 2 autres boucles de régulations (couple et vitesse). Il est possible à partir de cet écran de modifier le sens du retour de position.

B) Boucle de courant :

Icône :

Action : Permet de configurer la boucle courant du variateur



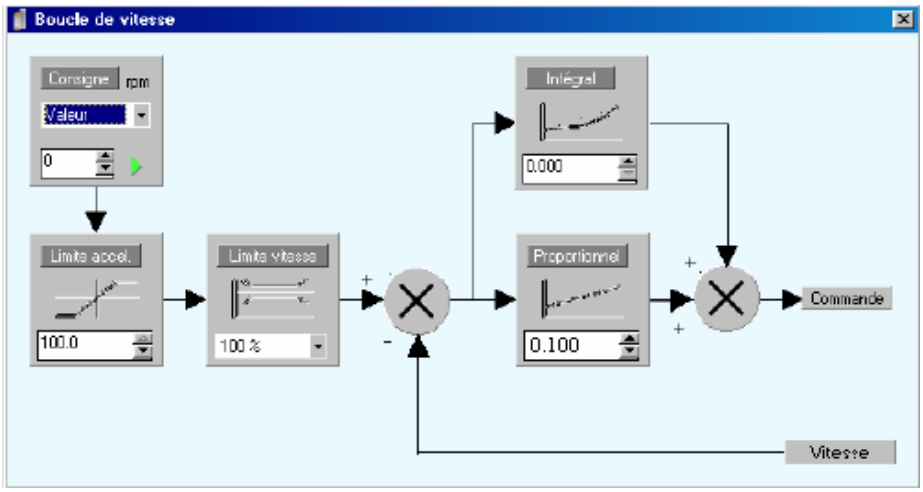
- Consigne : Sélection de la source (valeur, entrée analogique ou RS232) exprimée en pourcentage du courant maximal du moteur.
- Limite accélération : Limitation de la pente de la variation de courant
- Limite courant : Limitation du courant en pourcentage du courant maximal du moteur.
- Gain intégral : Régulation
- Gain proportionnel : Régulation

Les limites d'accélération et de courant sont accessibles en mode *paramètres avancés* (voir Menu / Options / Accessibilité)

C) Boucle de vitesse :

Icône :

Action : Permet de configurer la boucle vitesse du variateur



- Consigne : Sélection de la source : valeur, entrée analogique, RS232 …
- Limite accélération : Limitation de la pente de vitesse

Tableau de correspondance entre le pourcentage de la limite d'accélération et le temps pour passer de 0 à la vitesse nominale du moteur :

Pourcentage	Temps
100%	aucune limite
50%	20 ms
10%	100 ms
1%	1 s
0,10%	10 s

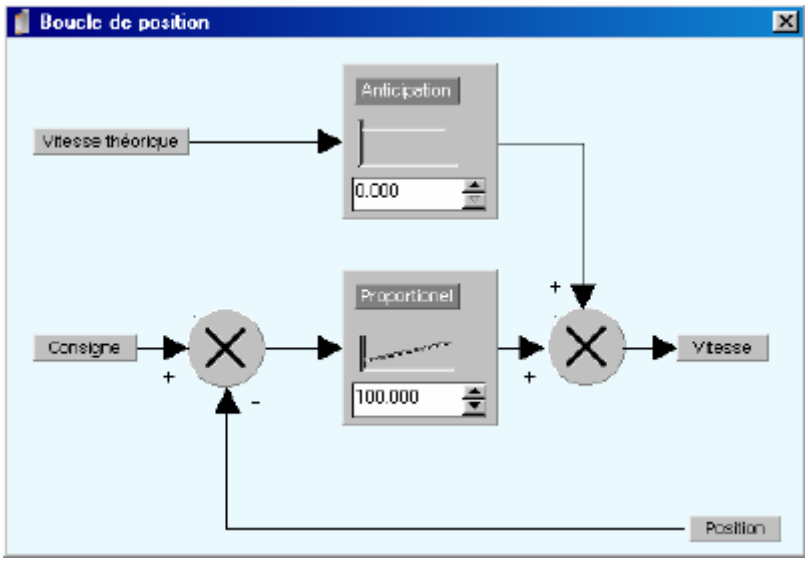
- Limite vitesse : Limitation de la vitesse en pourcentage de la vitesse nominale
- Gain intégral : Régulation
- Gain proportionnel : Régulation

Les limites d'accélération et de vitesse sont accessibles en mode *paramètres avancés* (voir Menu / Options / Accessibilité)

D) Boucle de position :

Icône :

Action : Permet de configurer la boucle de position du variateur.

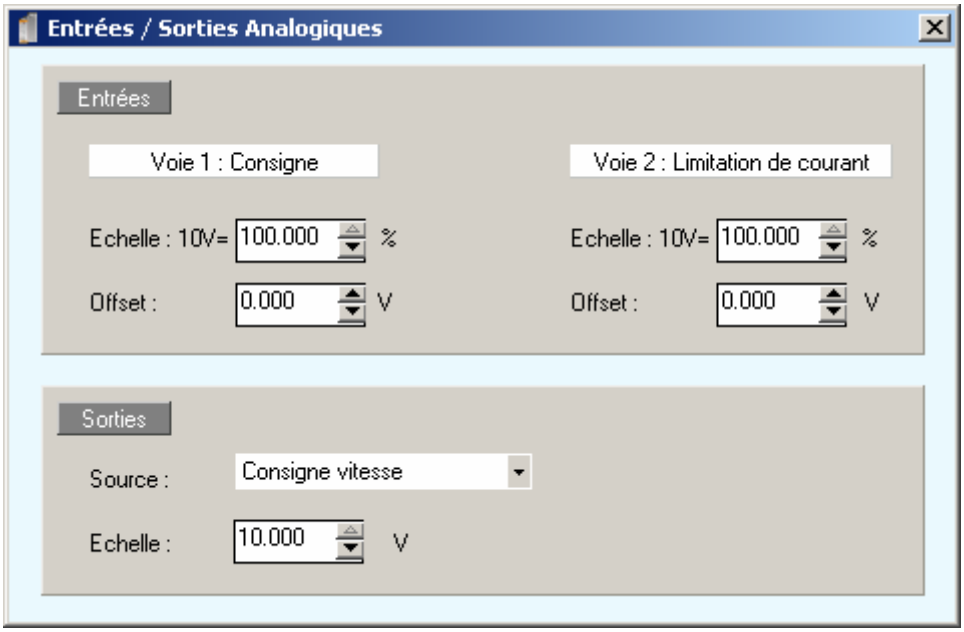


- Anticipation : Le gain d'anticipation de vitesse assure une erreur de poursuite proche de zéro.
- Gain proportionnel : Une valeur trop faible donne un asservissement mou, une valeur trop forte rend le système instable.

E) Entrées/Sorties analogiques :

Icône :

Action : Permet de configurer les entrées et sorties analogiques.



- Entrées analogiques :

  En mode boucle de courante, la voie 1 peut être utilisée comme consigne et la voie 2 en limite de courant avec comme valeur maximum : Inom * Imax (voir dans **Paramètres \ Moteur**)

  Echelle : 10V= : Permet d'associer un pourcentage pour 10V sachant que 100% correspond à la valeur maximale du courant ou de la vitesse.

  Ex : Vitesse nominale = 3000tr/min
  Vitesse maximale = 110 %
  Tension sur la voie 1 -> ± 5V

  On a alors la vitesse maximale du moteur à 3300 tr/min et on choisira une échelle de 200 % pour faire correspondre 5V à la vitesse maximale.

  Offset : Ajoute un offset à la valeur réelle reçue.

- Sortie analogique :

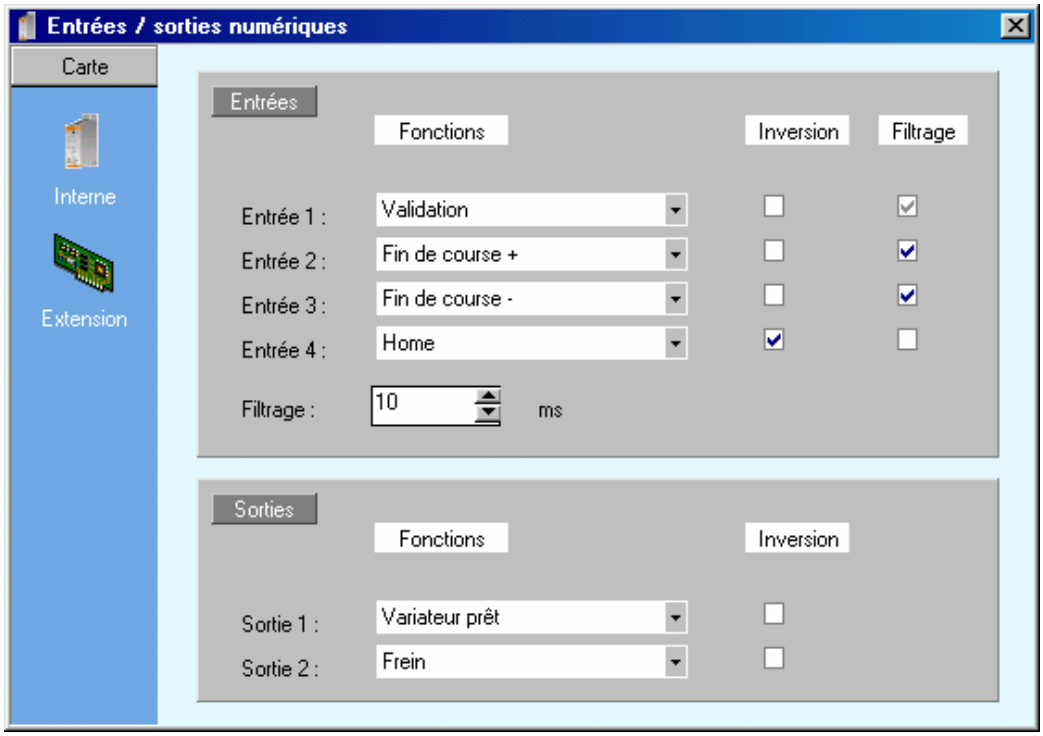
Consigne	Valeur min	Valeur max
Aucun	-	-
Position	- 1/2 tour	1/2 tour
Consigne de courant	- Inom.* Imax.	+ Inom.* Imax.
Courant mesuré	- Inom.* Imax.	+ Inom.* Imax.
Consigne vitesse	- Vit. nom. * Vit max.	+ Vit. nom. * Vit max.
Vitesse mesuré	- Vit. nom. * Vit max.	+ Vit. nom. * Vit max.
Erreur de poursuite	- Err. poursuite	+ Err. poursuite

Echelle : Permet de choisir la plage du signal de sortie.

F) Entrées/sorties digitales :

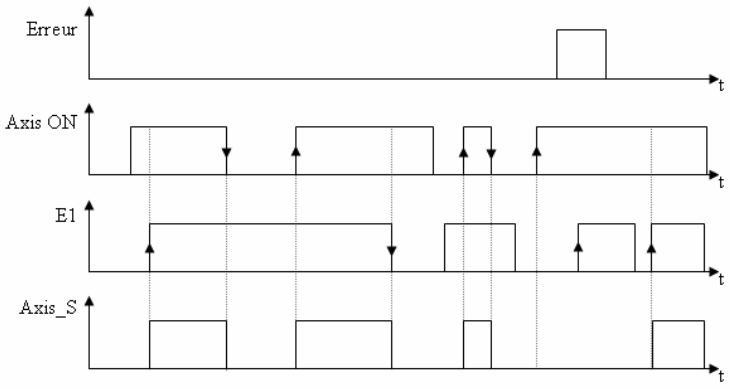
Icône :

Action : Permet de configurer les entrées et sorties logiques du variateur.

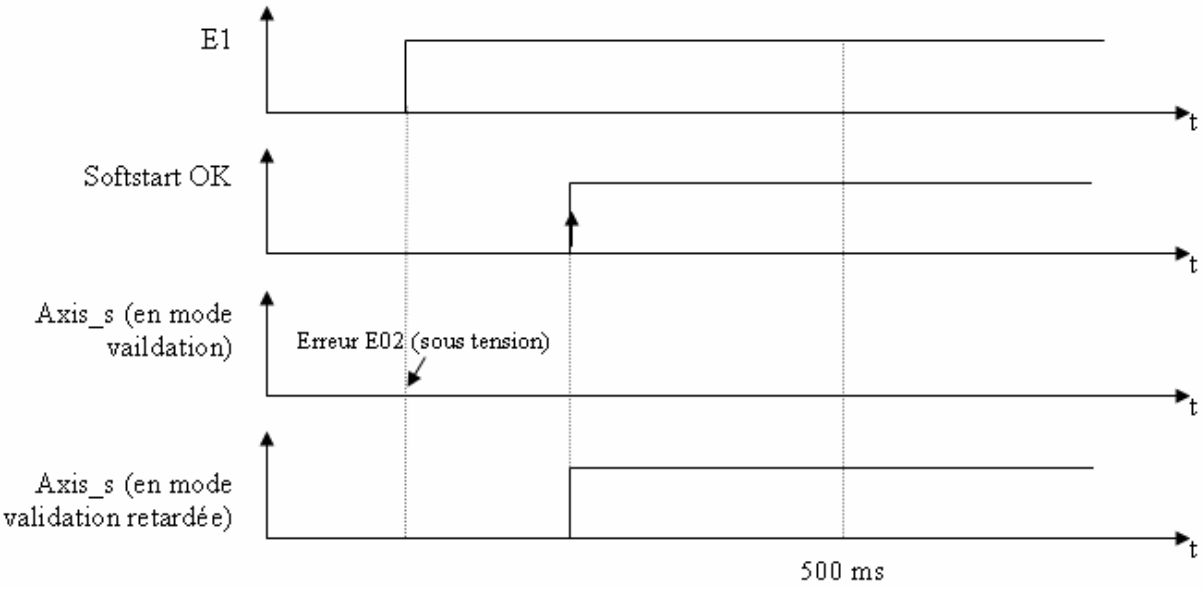


- Entrée 1 : Sélection : **Validation** du variateur ou aucune :

  1. Si **Aucune**, l'asservissement se fait par le bouton enable de la fenêtre principale du iDPL ou par l'instruction Axis on du langage iDPL.
  2. Si **Validation**, l'asservissement se fait sur front montant de l'entrée logique E1.
  3. Si **Validation + iDPL**, la demande d'asservissement se fait par front montant sur 1 des 2 conditions et niveau logique 1 sur la seconde:



4. Si **Validation retardée**, la demande d'asservissement se fait sur front montant de l'entrée logique E1 mais l'asservissement se fait sur validation du Softstart et validateur du codeur SINCOS (si utilisé), le délai maximum avant défaut est de 500ms :



- Entrée 2 : Sélection : **Fin de course +** ou aucune.
- Entrée 3 : Sélection : **Fin de course -** ou aucune (désactiver le filtrage pour fonctionner en entrée rapide).
- Entrée 4 : Sélection : **Capteur prise d'origine**, **Raz défaut** sur front descendant ou aucune (désactiver le filtrage pour fonctionner en entrée rapide).
- Délai de Filtrage : Valeur du filtre en ms.
- Inversion : Si inversion non activée, l'entrée est gérée en logique positive sinon en logique négative.
- Filtrage : Permet d'activer le filtrage sur l'entrée sélectionnée.
- Sortie 1 : **Variateur prêt** ou aucune.
- Sortie 2 : **Frein moteur** ou aucune

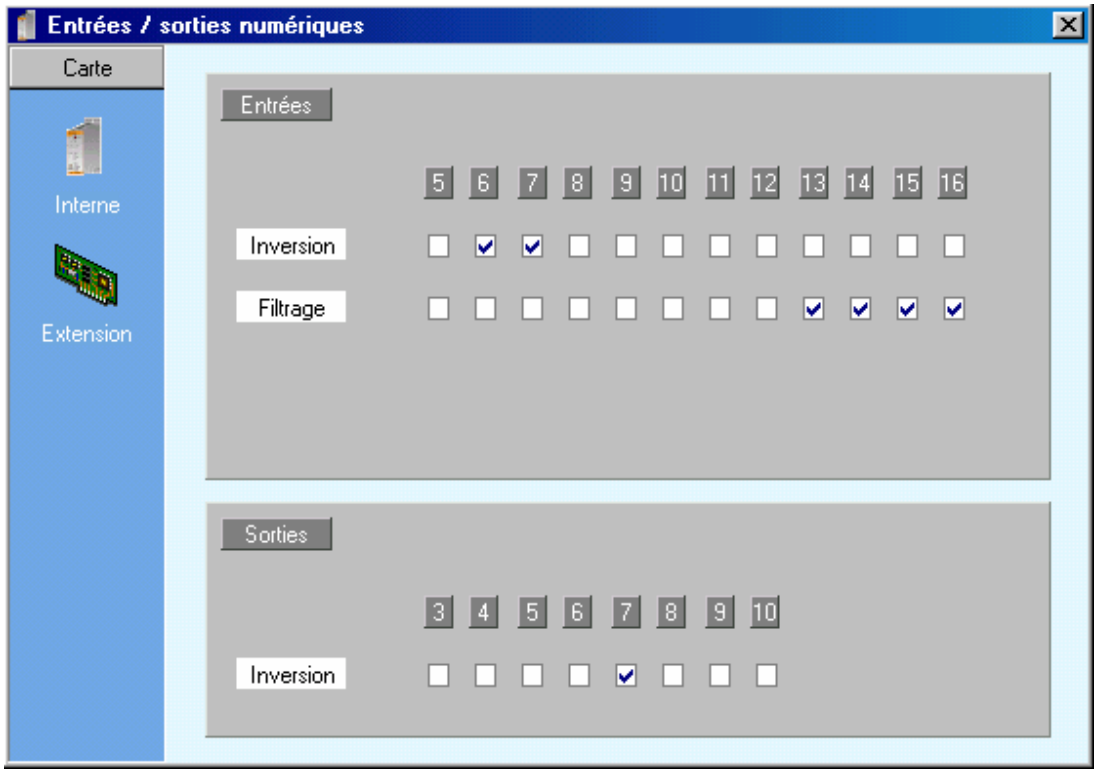
La sortie variateur prêt doit être insérée dans la boucle d'arrêt d'urgence.

Si le frein est sélectionné sur la sortie 2, il est nécessaire d'insérer un relais statique externe (la sortie n°2 étant limitée à 100 mA) avec une diode de roue libre.

L'état logique de la sortie frein correspond à l'état *enable* interne du variateur

La décélération urgente (motion control / profil de vitesse) est utilisée pour arrêter le mouvement lorsqu'on utilise les entrées de Fin de course quand le variateur est en mode position.

Pour utiliser les entrées rapides 3, 4, 15 et 16 en mode rapide, il faut désactiver leur filtrage.



En ajoutant une carte d'extension I/O, vous disposez de :

- 12 entrées supplémentaires pouvant être filtrées et/ou inversées (pour les entrées 15 et 16, désactiver le filtrage pour fonctionner en entrée rapide).
- 8 sorties supplémentaires pouvant être inversées.

G) Sécurités :

Icône :

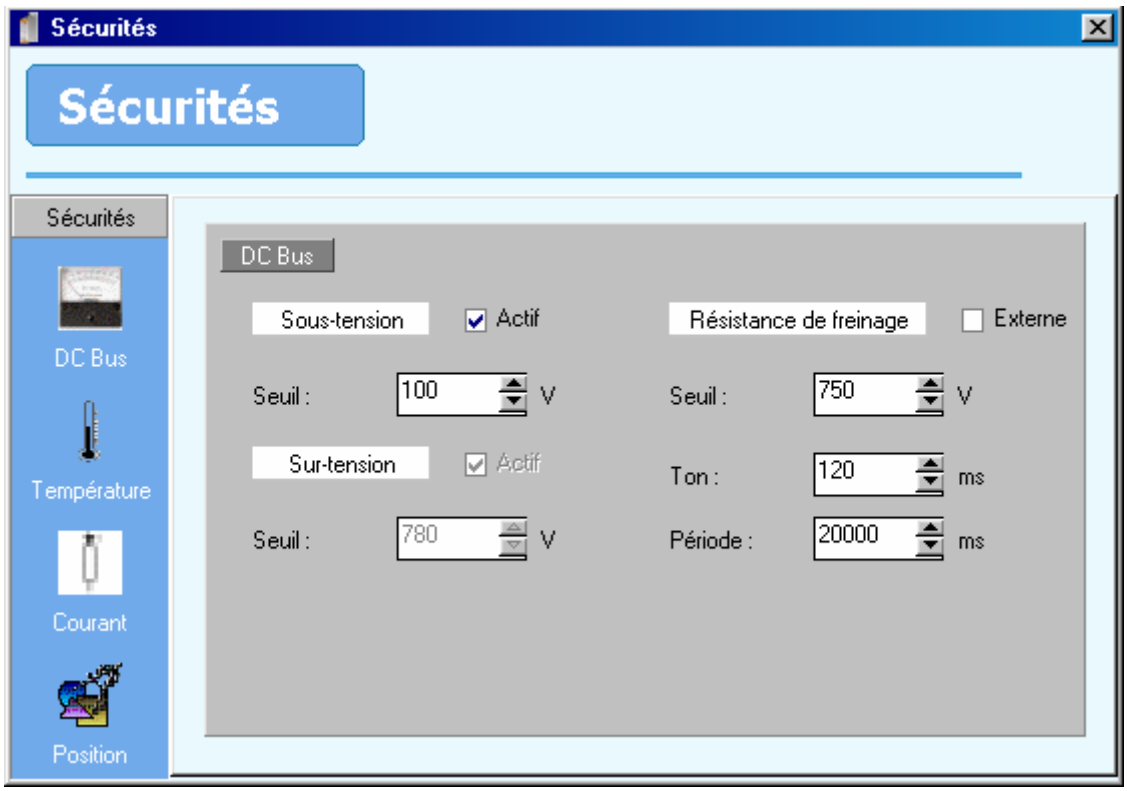
Action : Permet d'ajuster les paramètres de sécurité pour une sécurité maximale.

a) Sécurité DC Bus :

Réglages usines, ne pas les modifier.

Dans le cas où une résistance externe est nécessaire, cocher la case Résistance externe (si elle n'est pas cocher, le variateur utilisent des paramètres par défaut pour gérer le ballast).

cette résistance devra être bien dimensionnée sous peine de détérioration de celle-ci, son réglage est accessible en paramètres avancés (voir Menu /Options / Accessibilités).



- Sous-tension : actif par défaut, tension minimum du variateur lorsqu'il est asservi. Provoque le défaut E02 sous tension

- Sur-tension : actif par défaut, tension maximum du variateur. Provoque le défaut E01 sur-tension.

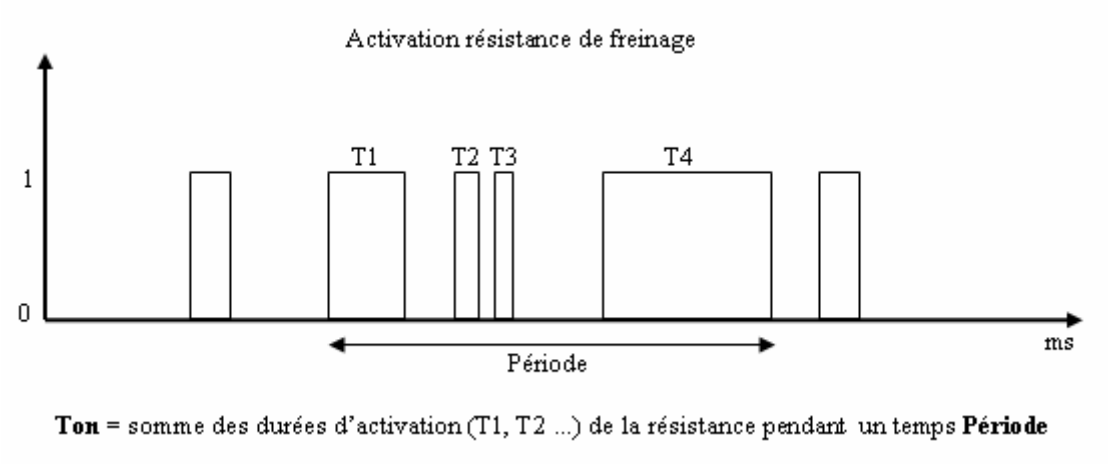
  Attention : Ce paramètre est modifiable seulement si on est en mode usine et si le paramètre **Tension nominal** est sur « **Autre** » (fenêtre paramètres, onglet variateur) sinon une valeur par défaut lui est attribuée (390 pour un drive 230V, 780 pour un drive 400V).

- Résistance de freinage externe : à activer que si une résistance externe a été connectée au variateur.

- Seuil de freinage : permet de définir à partir de quelle tension, le sur-tension DCBus sera dissipé par la résistance de freinage.

  Attention : Ce paramètre est modifiable si on est en mode usine et si le paramètre **Tension nominal** est sur « **Autre** » (fenêtre paramètres, onglet variateur) sinon une valeur par défaut lui est attribuée (375 pour un drive 230V, 750 pour un drive 400V).

- Ton et Période : permettent de définir la durée d'activation de la résistance de freinage :



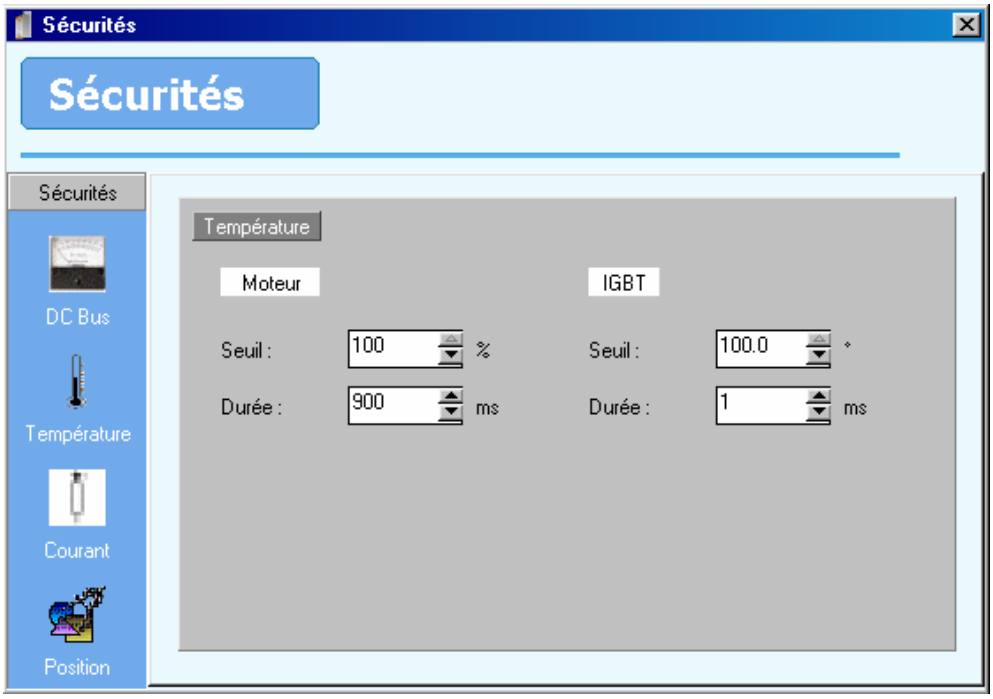
Ton = somme des durées d'activation (T1, T2 ...) de la résistance pendant un temps **Période**

Tant que la durée Ton n'est pas dépassée la surtension DCBus peut être dissipé par la résistance de freinage après le variateur passe en défaut E01 sur-tension.

Attention : Ces paramètres sont utilisés que si la résistance de freinage externe est activée sinon des valeurs par défaut leurs sont attribuées.

b) Sécurité température :

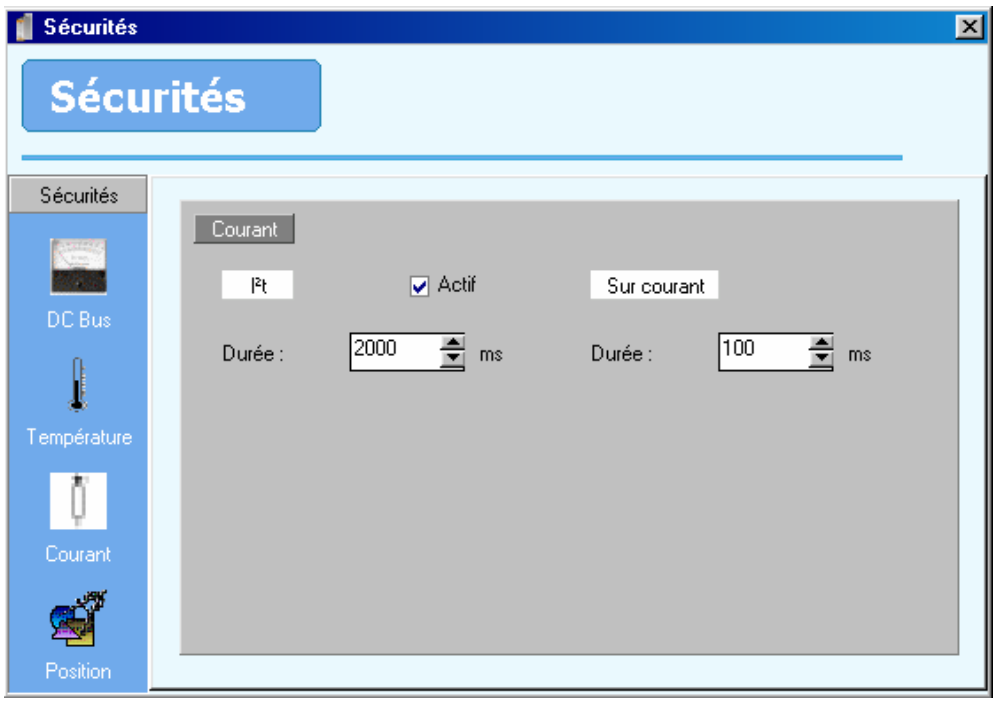
Réglages usines, ne pas les modifier.



- Température moteur : permet de définir le seuil et la durée de dépassement de température moteur pour provoquer le défaut E07.
- Température IGBT : permet de définir la température seuil et la durée de dépassement du module IGBT pour provoquer le défaut E06.

c) Sécurité courant :

Réglages usines, ne pas les modifier.



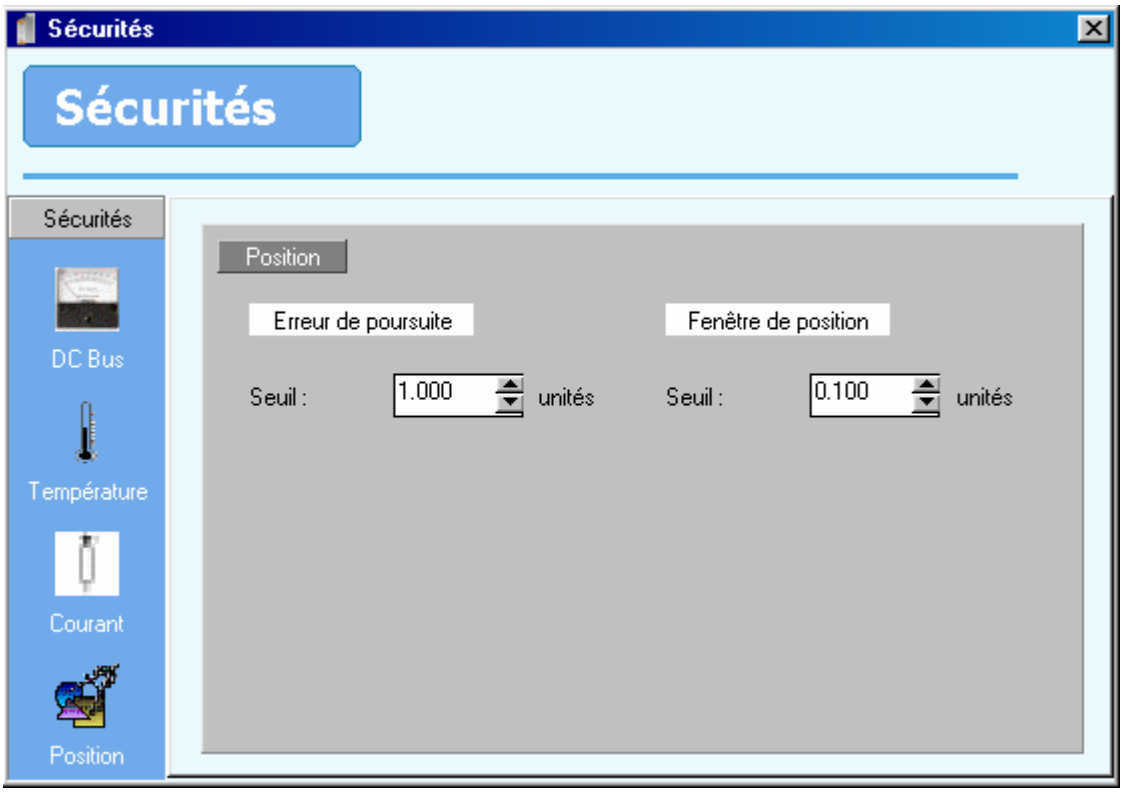
- I²t : Les moteurs brushless acceptent des courants crêtes (jusqu'à 2 fois le courant nominal). I2t permet de surveiller le courant moyen du variateur, sachant que celui-ci ne doit pas dépasser le courant nominal. En fonctionnement stabilisé, I2t doit rester à 0.

  La durée permet de définir la période sur laquelle le contrôle du courant moyen est effectué.

- Sur-courant : Le variateur contrôle en permanence si le courant est dans une certaine plage (dépend du type de variateur), si ce courant est hors limite pendant durée du paramètre sur-courant alors le variateur passe en défaut E04 sur-courant

d) Sécurité position :

Lorsque le variateur est utilisé en mode position, régler le seuil d'erreur de poursuite au minimum. Attention, la valeur maximale admissible est de 20 tours moteurs. La valeur de ce seuil doit être la plus faible possible, par exemple 0,2 tour moteur.



- Erreur de poursuite : Dès que l'axe passe en mode asservi, il est contrôlé à tout moment : à l'arrêt, en mouvement. Si la différence entre sa position théorique calculée et sa position réelle donnée par le retour codeur est supérieure à l'erreur de poursuite maxi, le variateur passe l'axe servo en mode non asservi.

  Le réglage de cette valeur est très importante : une valeur trop petite entraîne des arrêts intempestifs sur l'axe, une valeur trop grande influe sur la sécurité des organes électriques et mécaniques.

  Attention : la valeur de l'erreur de poursuite se fait dans l'unité sélectionnée et dépend des paramètres de l'écran **Motion control \ Unités**.

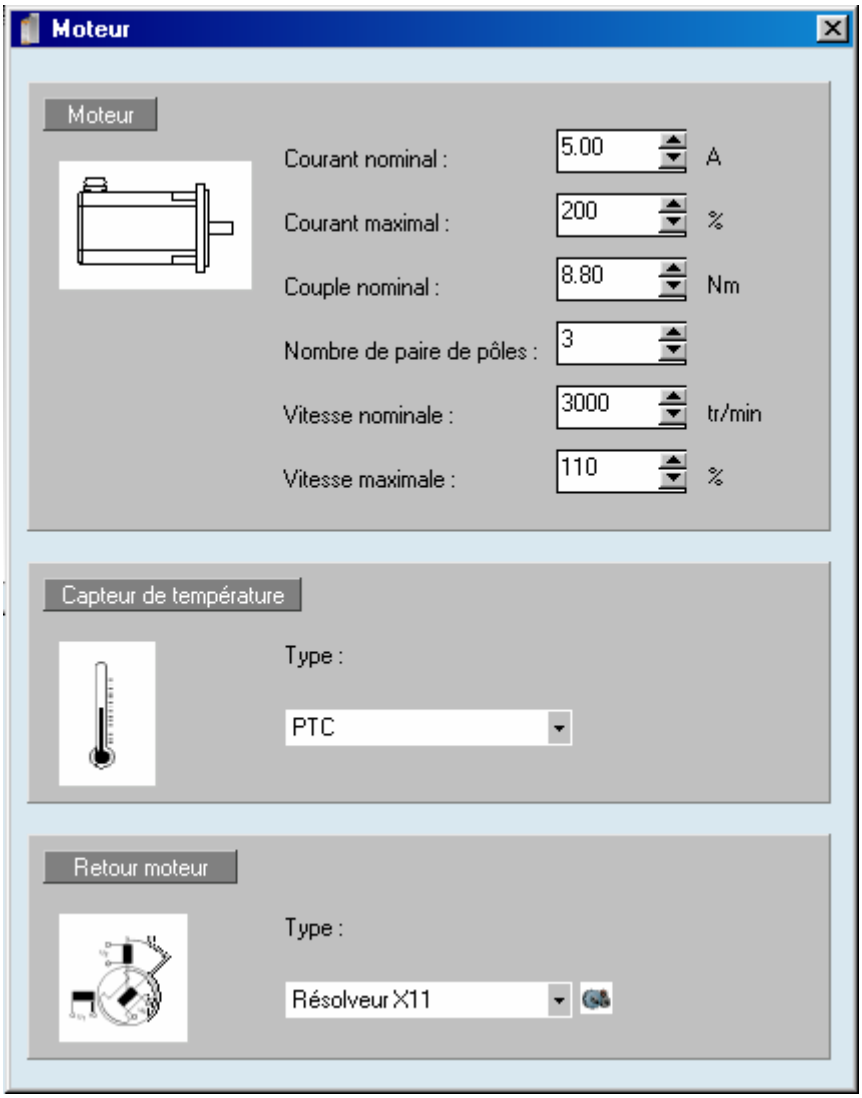
- Fenêtre de position : Ce paramètre est utilisé pour modifier la fenêtre de positionnement minimale entre la position réelle et la position théorique. Après un déplacement, si la différence entre la position réelle et la position demandée sont inférieurs à la fenêtre de position, le système considère que la position est atteinte.

  Attention : la valeur de la fenêtre de position se fait dans l'unité sélectionnée et dépend des paramètres de l'écran **Motion control \ Unités**.

H) Moteur :

Icône :

Action : Permet de configurer le moteur et le résolveur.


Moteur
Moteur
Courant nominal :
5.00
A
Courant maximal :
200
%
Couple nominal :
8.80
Nm
Nombre de paire de pôles :
3
Vitesse nominale :
3000
tr/min
Vitesse maximale :
110
%
Capteur de température
Type :
PTC
Retour moteur
Type :
Résolveur X11

a) Moteur :

Courant nominal : Courant nominal du moteur en A.

Courant maximal : Pourcentage par rapport au courant nominal. Par défaut 200% (Imax = 2 * Inom).

Couple nominal : Couple nominal du moteur en Nm. Cette information n'est pas utilisée par la régulation et est juste à titre indicatif.

NB paire de pole : Suivant le type de moteur (faire un autotunning résolveur).

Vitesse nominal : Vitesse nominal du moteur en tr/mn

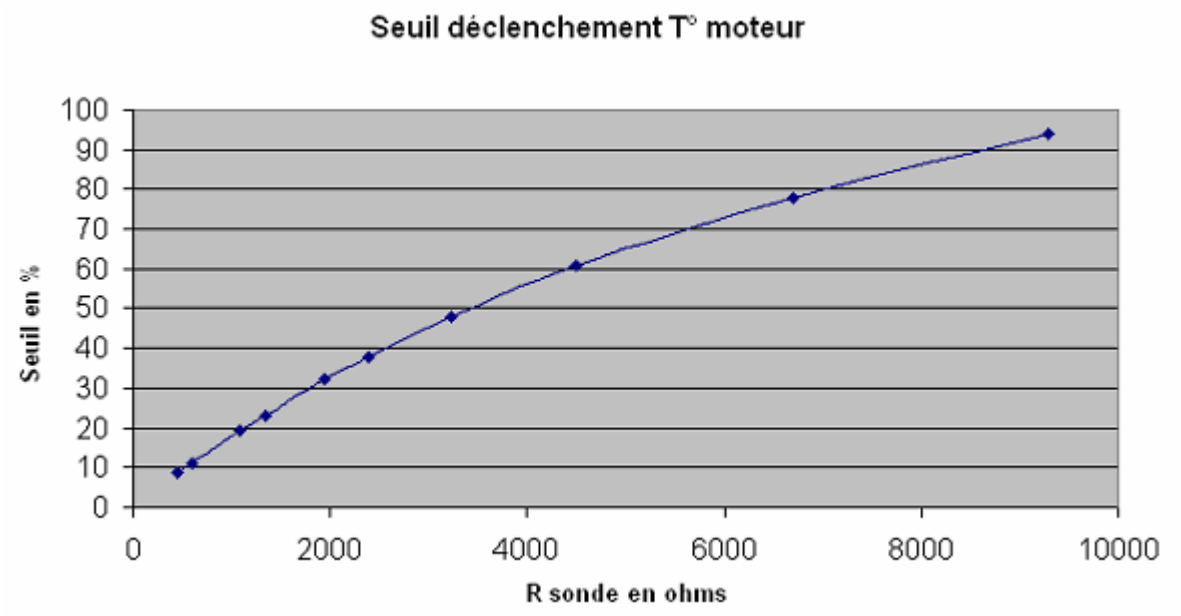
Vitesse maximal : Pourcentage de la vitesse nominal utilisé dans le boucle de vitesse pour limiter le vitesse du moteur.

b) Capteur de température :

Type : Réglage usine (PTC ou NTC).

Sonde PTC : l'erreur est déclenchée lorsque la résistance de la sonde est supérieure au seuil du variateur.

Sonde NTC : l'erreur est déclenchée lorsque la résistance de la sonde est inférieure au seuil du variateur.



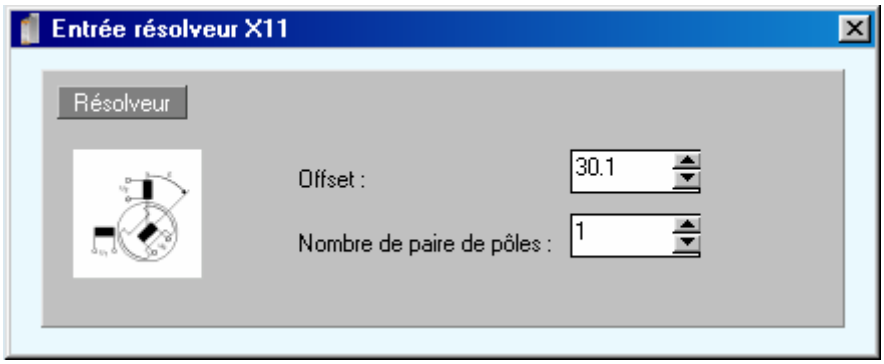
c) Retour moteur :

Type : Choix du retour moteur : retour résolveur X11 ou retour SinCos X13.

I) Résolveur :

Icône :

Action : Permet de configurer le retour résolveur du variateur



Offset : calage résolveur.

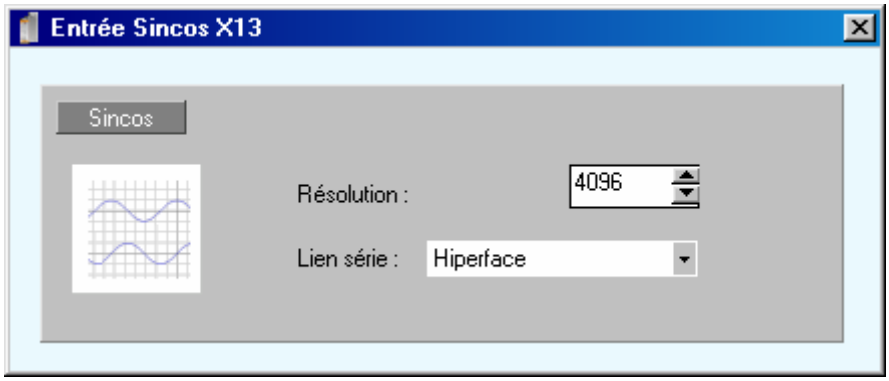
NB paire de pole : 1 paire de pôles pour la plus part des moteurs

Pour un résolveur ayant plusieurs paires de pôles, la position du rotor sur un tour pourra être décalée de NB paire de pôles / 360° (ex : 0, 120° ou 240° pour un résolveur avec 3 paires de pôles). De la même manière, une prise d'origine sur TOP Z pourra être décalé de NB paire de pôles / 360°.

J) SinCos :

Icône :

Action : Permet de configurer le retour SinCos du variateur



Résolution : Entrer la résolution en nombre d'incréments (4 incréments par point). Ex : pour un codeur 500 points rentrer 2000 incréments.

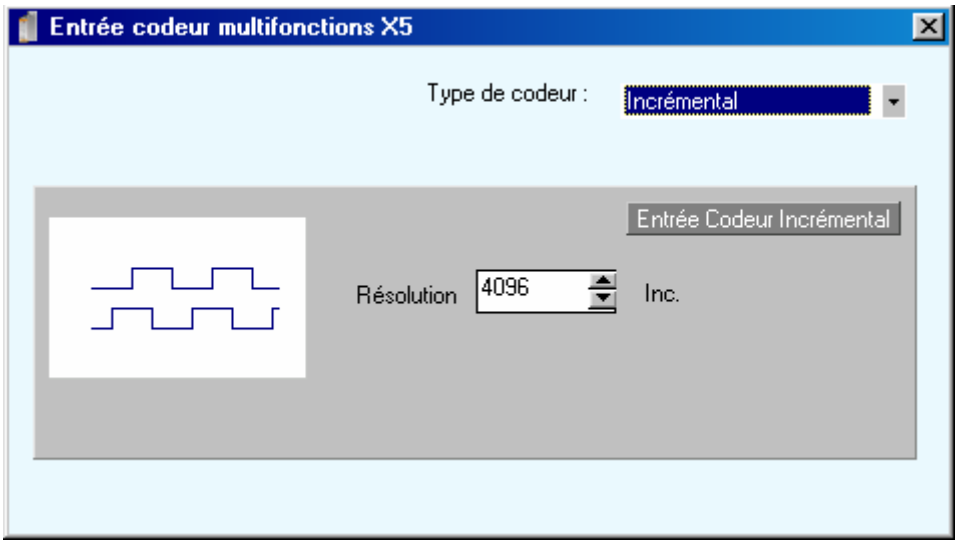
Lien série : si aucun est sélectionné, alors le retour position est relatif ; si Hiperface est sélectionné, on reçoit une position absolue.

En cas de défaut sur le lien série, une erreur résolveur E08 ce produira lors de la demande d'asservissement.

K) Entrée codeur multifonctions:

Icône :

Action : Permet de paramétrer l'entrée codeur

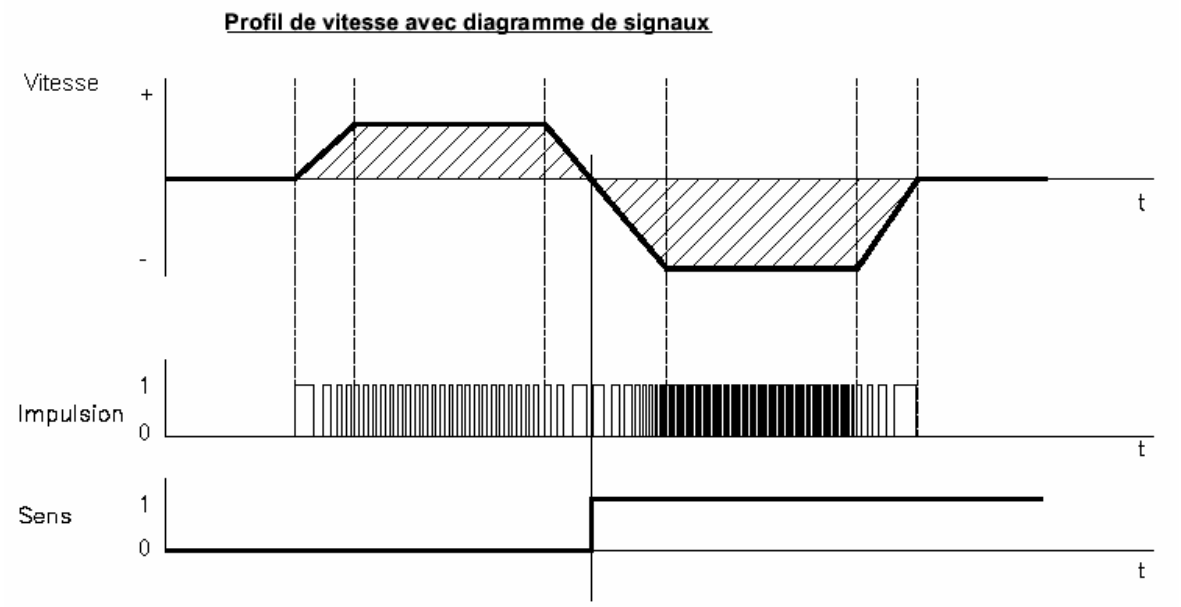


a) Mode incrémental :

Résolution : Codeur maître : rentrer la résolution en nombre d'incréments (4 incréments par point). Ex : pour un codeur 500 points rentrer 2000 incréments.

b) Mode Stepper :

Permet de raccorder une commande de moteur pas à pas d'un constructeur quelconque au variateur iMD. Le nombre de pas est réglable ainsi que le sens de rotation.

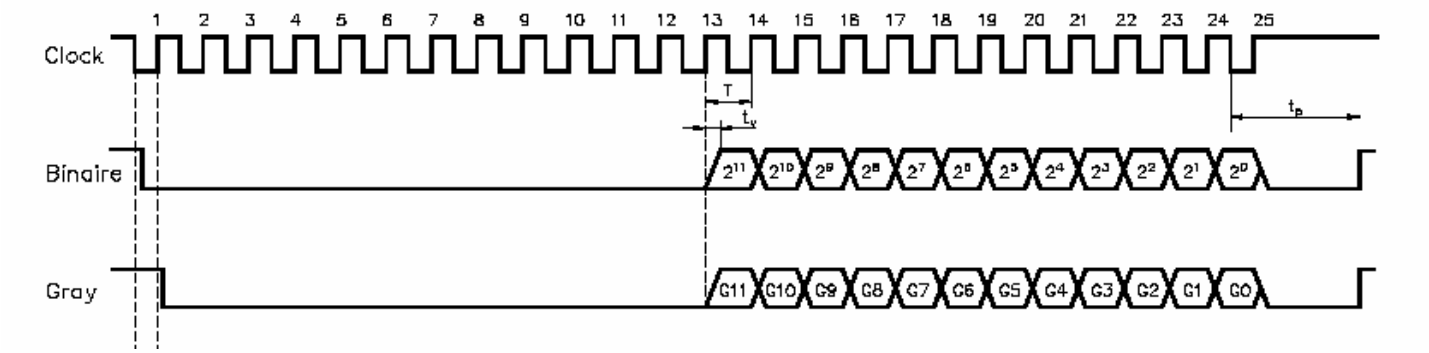


Résolution : Codeur maître : rentrer la résolution du codeur STEPPER en nombre d'incréments.

c) Mode SSI :

Permet de raccorder un codeur absolu de type SSI pour les fonctions codeur maître ou régulation double boucle (retour boucle de position).

C'est à partir des signaux cycliques absolus sur l'entrée codeur que la position de l'arbre moteur est calculée.



Bit : Nombre de bit composant l'information de la position (de 2 à 31).

Fréquence : Fréquence d'horloge Clock (1,5 Mhz maxi)

Résolution : Codeur maître : entrer la résolution du codeur, en nombre d'incréments.

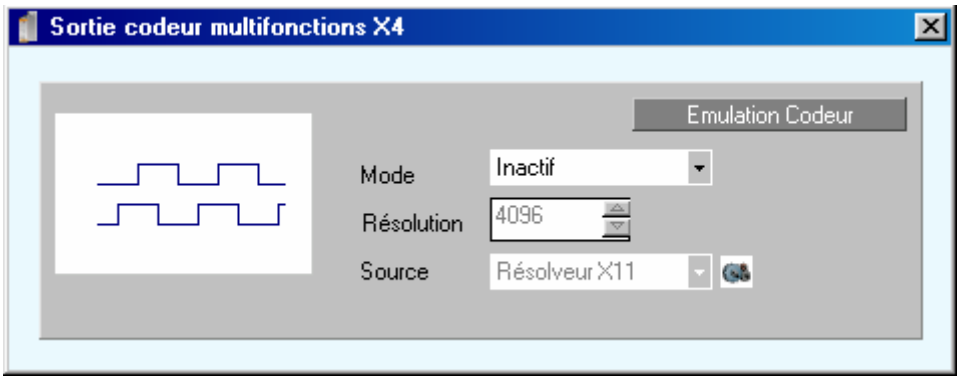
Code GRAY: Oui/Non (dépend du type de codeur)

Attention : La résolution doit être inférieure ou égale à 2^nb Bit et la durée maxi pour un échantillonnage (2^nb Bit / Fréquence) doit être inférieure à 100µs.

L) Sortie codeur multifonctions :

Icône :

Action : Permet de paramétrer la sortie codeur



Mode inactif: La sortie codeur n'est pas utilisée.

Mode actif : La sortie codeur renvoie un signal incrémental par rapport à la source sélectionné et la résolution saisie.

- Source : Résolveur, SinCos, Entrée multifonction (incrémental, stepper, SSI, IMD bus), Virtuel, Analogique
- Résolution : la résolution de la sortie codeur en nombre d'incréments.

Mode bypass: (codeur incrémental) L'entrée codeur multifonctions est recopiée sur la sortie codeur.

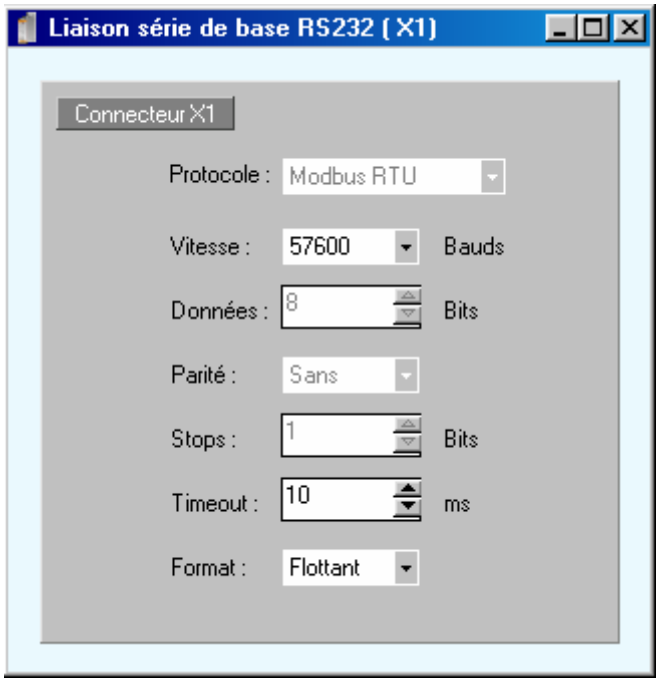
M) Liaison RS232 de base :

Icône :

Action : Permet de paramétrer la communication Modbus.

Le variateur gère cette liaison en Modbus RTU esclave.

Le format 8 bits de données, 1 bit de stop, pas de parité, est figé.



Dans cette fenêtre, on paramètre la vitesse de transmission et le timeout dans le cas où l'on est pas en « communication système ». Lorsque l'on utilise cette liaison en « communication système » (réglage par défaut à partir du menu Options / ComPC), la vitesse est figée à 57600 bauds.

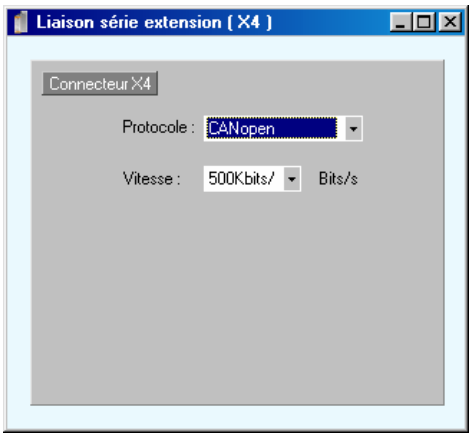
En communication système, le signal RTS du PC est utilisé et forcé à l'état logique 1.

N) Liaison d'extension :

Icône :

Action : Permet de paramètrer la liaison d'extension en CANopen, RS232, RS422 ou RS485.

- CANopen :



Vitesse : permet de définir la vitesse de communication sur le bus CANOpen.

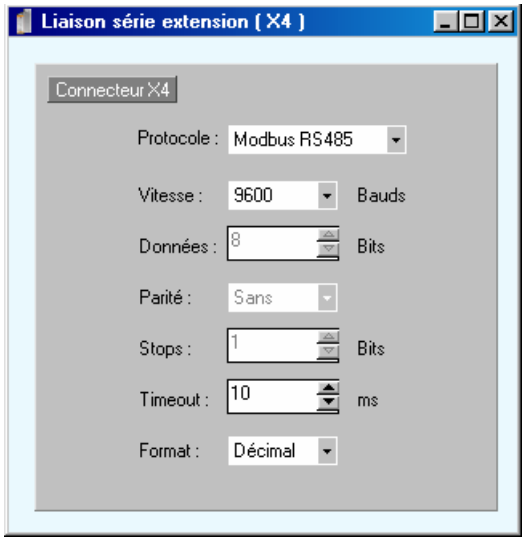
Pour plus de renseignements, voir les annexes sur le bus CANOpen .

Choisir un numéro de Node ID dans l'écran principal pour communiquer avec le variateur associé sur le bus CANopen.

- Port RS232, RS422 ou RS485 :

Le variateur gère cette liaison en Modbus RTU esclave.

Le format 8 bits de données, 1 bit de stop, pas de parité, sont figés.



Protocole : permet de choisir le support de la liaison.

Le NodeID du variateur correspond aux 5 premiers dipswitchs + 1 de la carte de communication.

Ex: dipswitchs: 1 -> ON, 2 -> OFF, 3 -> ON, 4 -> OFF, 5 -> OFF

Dipswitchs value = 1 + 4 = 5

NodeID = 5 + 1 = 6

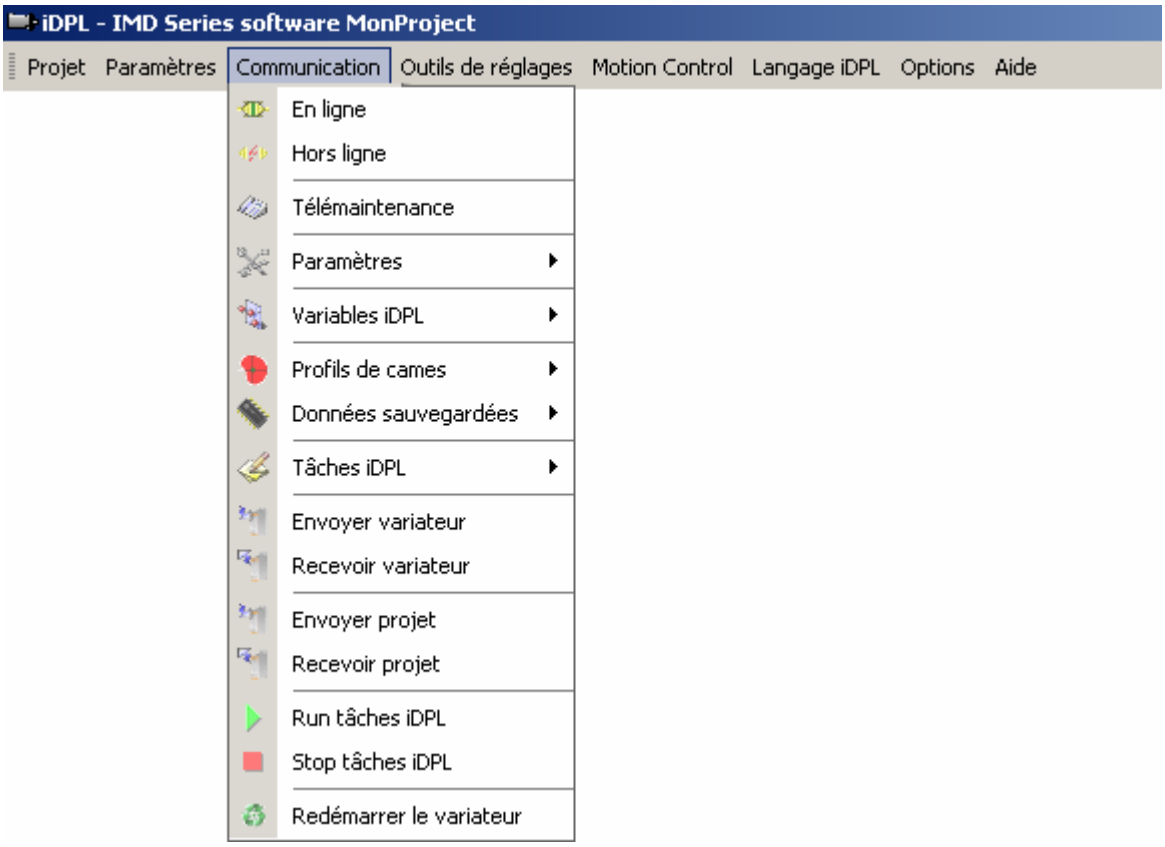
Vitesse : permet de définir la vitesse de communication du port.

TimeOut : temps maximum de non réponse.

Format : permet de choisir le format des réels (variables et paramètres)

- Flottant : utiliser en communication système
- Décimal : format le plus répandu, le nombre de chiffre après la virgule dépend du paramètre précision dans options\langage iDPL\Compilateur.

3-3-3- Communication



A) En ligne :

Icône :

Action : Permet d'établir la liaison avec le variateur. Tous les paramètres affichés sur l'écran du PC correspondent aux valeurs stockées dans le variateur.

B) Hors ligne :

Icône :

Action : Permet de travailler sans être relié au variateur.

C) Paramètres :

Icône :

Action : Si votre variateur communique avec le logiciel, vous pourrez :

- ***Envoyer les paramètres PC -> Variateur*** : permet d'envoyer un fichier de paramètres du PC vers le variateur. Ces paramètres sont automatiquement sauvés dans le variateur.
- ***Importer depuis un fichier et envoyer :*** permet d'envoyer un fichier de paramètres extérieur au projet vers le variateur. Ces paramètres sont automatiquement sauvés dans le variateur.
- ***Sauvegarder les paramètres variateur*** : permet d'enregistrer les paramètres courants du variateur dans sa mémoire Flash pour en assurer la sauvegarde même après mise hors tension du variateur (coupure de l'alimentation 24VDC du variateur).

D) Trajectoires :

Icône :

Action : Permet d'envoyer ou de recevoir les 64 profils de trajectoires préenregistrés.

E) Variables iDPL :

Icône :

Action : Permet d'envoyer ou de recevoir toutes les variables du variateur.

F) Profile de came :

Icône :

Action : Permet d'envoyer, recevoir des profiles de came en FRAM.

Pour recevoir les profils de came d'un variateur, chaque début de table et nombre de points de came doit avoir été configurés avant.

G) Données sauvegardées :

Icône :

Action : Permet d'envoyer, recevoir les données sauvegardées en FRAM.

H) Tâches iDPL :

Icône :

Action : Permet d'envoyer, recevoir ou d'effacer les tâches du variateur.

I) Envoyer variateur :

Icône :

Action : Permet de faire un envoie groupé vers le variateur : il est possible de sélectionner les paramètres, les variables, les cames et les tâches.

J) Recevoir variateur :

Icône :

Action : Permet de faire une réception groupé du variateur : il est possible de sélectionner les paramètres, les variables, les cames et les tâches.

K) Envoyer projet :

Icône :

Action : Permet de faire un envoie groupé vers tous les variateurs du projet : il est possible de sélectionner les paramètres, les variables, les cames et les tâches.

L) Recevoir projet :

Icône :

Action : Permet de faire une réception groupé de tous les variateur du projet : il est possible de sélectionner les paramètres, les variables, les cames et les tâches.

M) Run iDPL :

Icône :

Action : Permet de démarrer le iDPL. Le variateur exécute toutes les tâches activées et ayant un démarrage automatique.

N) Stop iDPL :

Icône :

Action : Permet d'arrêter le iDPL. Toutes les tâches s'arrêtent.

O) Redémarrer :

Icône :

Action : Permet de redémarrer le variateur.

3-3-4- Outils de réglages

A) Tableau de bord :

Icône :

Action : Grâce à un ensemble d'outils, le tableau de bord permet de faire des controles et diagnostic rapides :

a) Permet de visualiser l'état du variateur et du moteur :

- Le voyant ILimit indique que le variateur est en limite de courant et que l'erreur de poursuite augmente à ce moment.
- Degrés position : indique la position du moteur en modulo 360°.
- Vitesse : indique la vitesse du moteur en tour par minute.

b) Permet de visualiser l'état des E/S analogiques et modifier la sortie

c) Permet de visualiser l'état des E/S numériques et modifier les sorties :

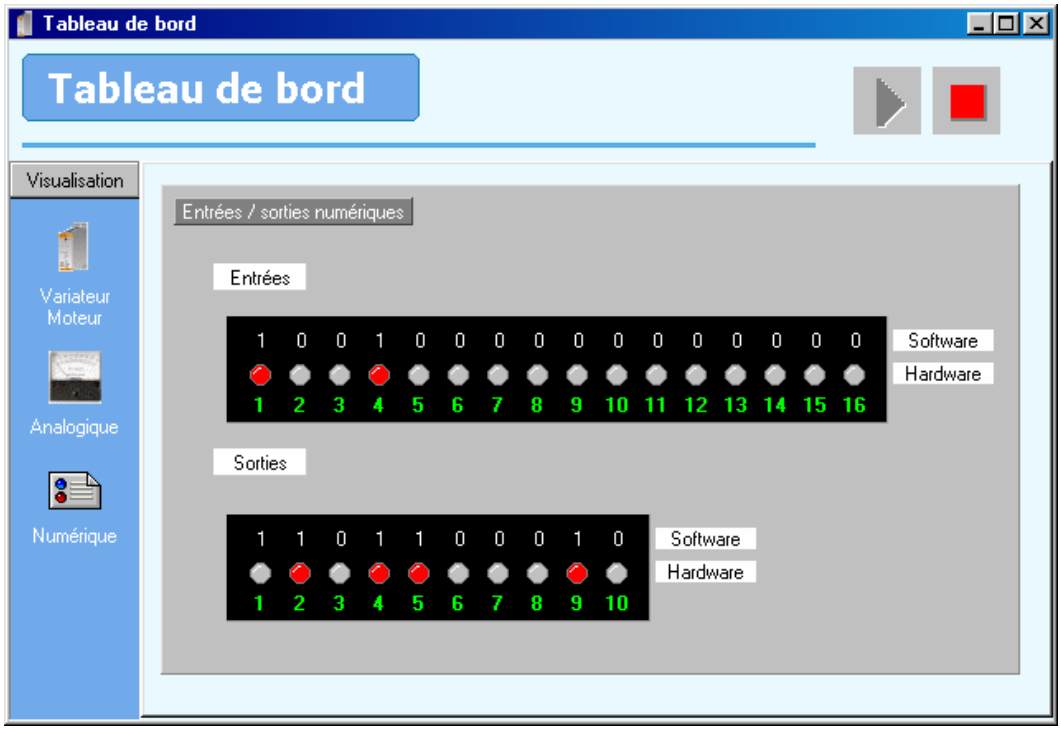


Tableau de bord
Tableau de bord
Visualisation
Variateur Moteur
Analogique
Numérique
Entrées / sorties numériques
Entrées
1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Software
Hardware
Sorties
1 1 0 1 1 0 0 0 1 0
1 2 3 4 5 6 7 8 9 10
Software
Hardware

Pour modifier l'état d'une sortie, il suffit de cliquer sur le bouton au dessus du numéro de la sortie, les numéros de sorties affichés en rouge indiquent que la sortie n'est pas modifiable car une fonction a été attribuée à cette dernière (variateur prêt, frein ...).

B) Afficher les défauts :

Icône :

Action : Permet de visualiser les défauts du variateur

En cas de défaut, une dévalidation et revalidation du variateur (entrée E1 ou bouton enable dans l'écran principal du logiciel ou par l'instruction Axis off / Axis on du langage iDPL) efface les défauts.

C) Autotuning :

Icône :

Action : Réalise un calage automatique entre le résolveur et le moteur, paramétrage automatique des différentes boucles de régulation.

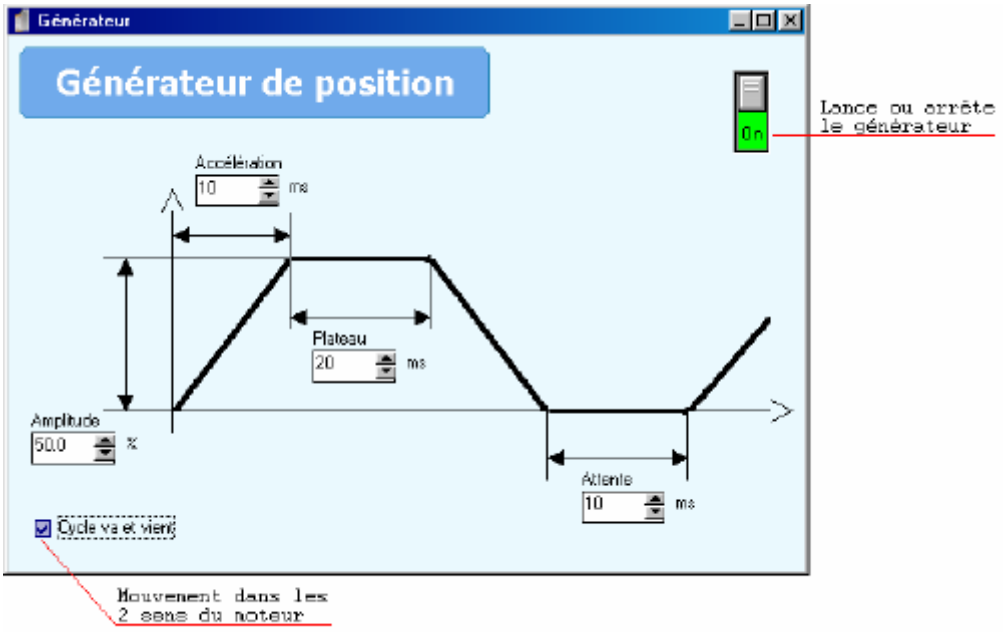
Voir chapitre auto tuning des boucles de régulations

Option disponible seulement si les paramètres avancés sont activés

D) Générateur :

Icône :

Action : Permet de lancer différents types de trajectoires pour optimiser les tests des boucles d'asservissements.



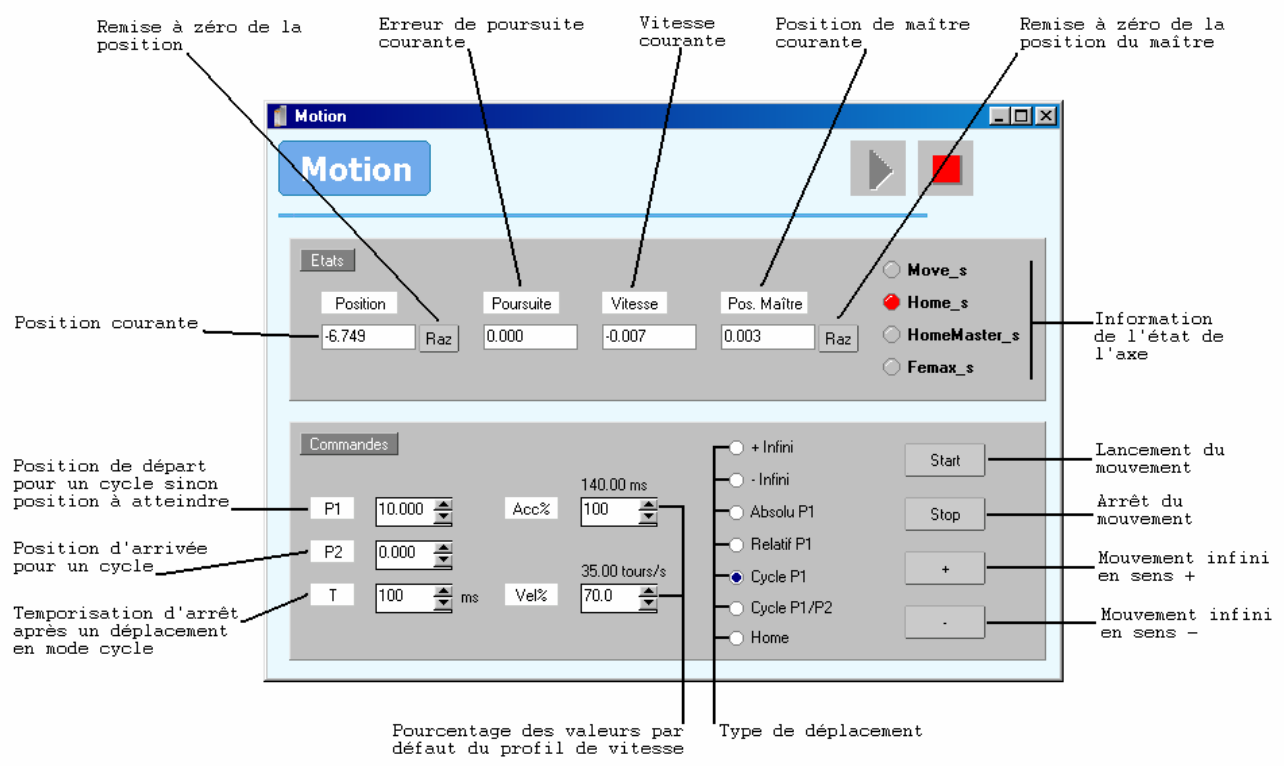
- Configurer le générateur pour effectuer le mouvement désiré.

- Asservir le moteur avec le bouton ENABLE (et/ou l'entrée E1 validation variateur)
- Lancer le mouvement avec le bouton ON/OFF du générateur

E) Motion :

Icône :

Action : permet de tester **la boucle de positionnement** de l'axe. Il est préférable de commencer par vérifier le comportement du moteur/variateur en forçant la consigne à une valeur comprise entre +10V et –10V (L'axe doit être en mode débrayé). On peut ensuite passer en mode asservi et régler les paramètres d'asservissement. Si l'on souhaite sauvegarder ces modifications, il faut faire une sauvegarde des paramètres dans le variateur.

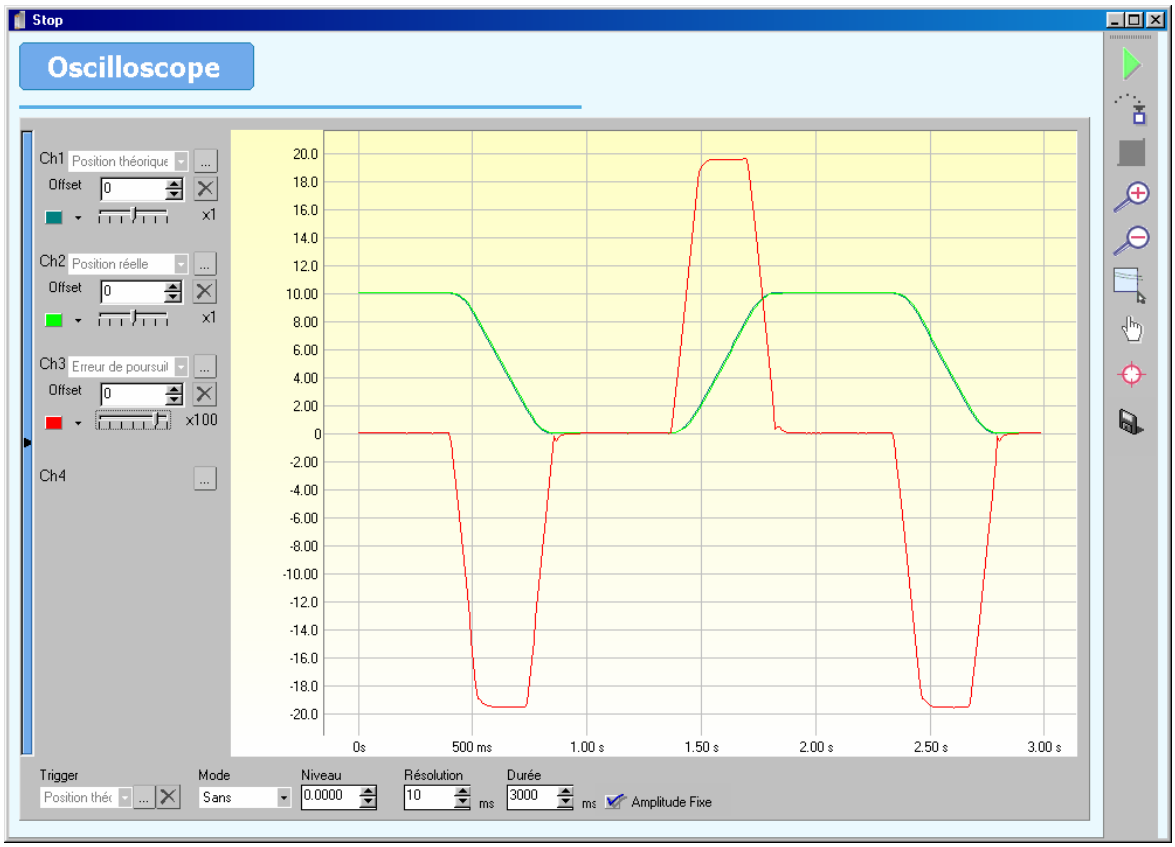


F) Oscilloscope :

Icône :

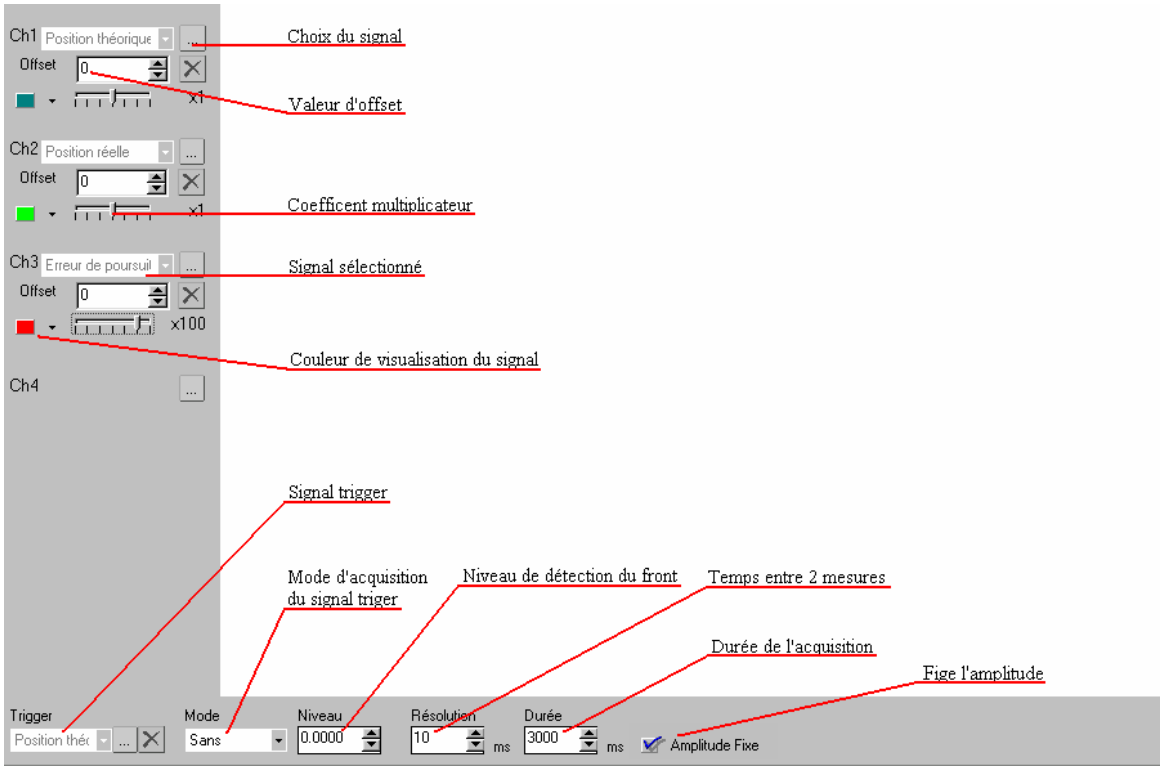
Action : Cette commande ouvre l'oscilloscope. Cet outil d'aide à la mise en œuvre permet de visualiser toutes les informations du variateur. Il est capable d'enregistrer jusqu'à 4 signaux simultanément.

L'oscilloscope est configuré en trois parties : l'écran de visualisation, la zone de configuration de l'acquisition, zone de réglage de la visualisation.



↳L'écran de visualisation est la partie centrale de l'oscilloscope où sont affichées les courbes.

↳La zone de configuration de l'acquisition permet de choisir les signaux à acquérir et de configurer le mode d'acquisition: le nombre d'échantillon, durée …

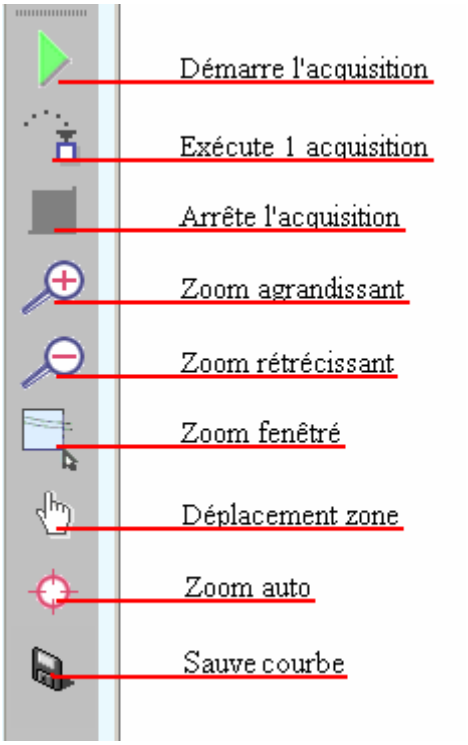


Chaque signal est affiché dans son unité

Exemple : courant en ampère, vitesse en tours/minute

Le coefficient d'un canal permet d'augmenter ou réduire l'amplitude du signal à l'affichage.

↳La zone de réglage de la visualisation permet de lancer ou arrêter l'acquisition et de modifier l'affichage de l'écran de visualisation.



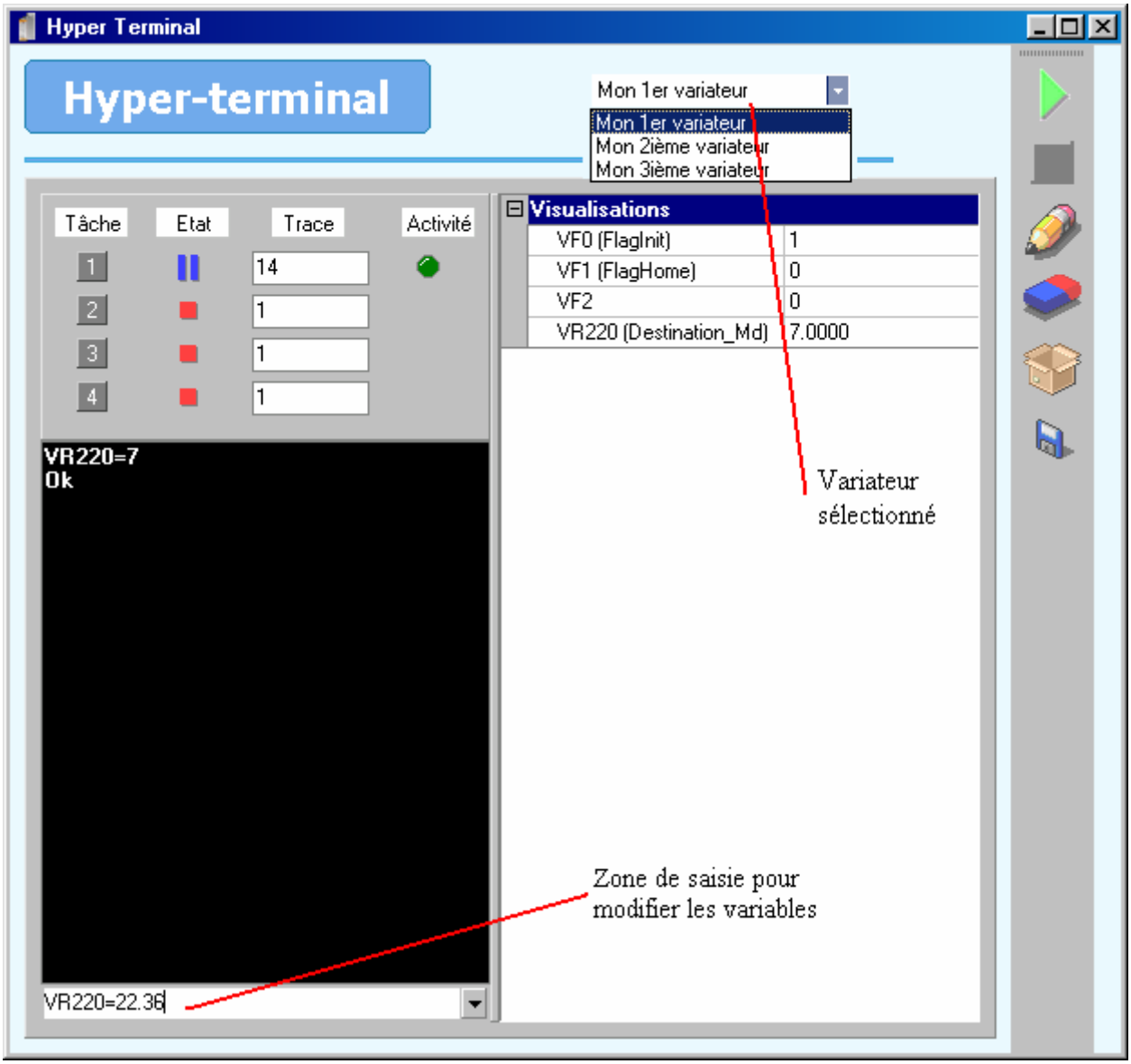
- Zoom fenêtré : Cliquer sur le bouton zoom fenêtré, le bouton devient actif puis tracer un rectangle dans le zone de visualisation des courbes en restant appuyer sur le bouton gauche de la souris, relâcher le bouton gauche pour valider le zoom.
- Sauve courbe : Permet d'enregistrer l'acquisition réalisée en fichier HTML et JPEG

G) Hyper terminal :

Icône :

Action : Cette commande ouvre l'hyper terminal. Cet outil d'aide à la mise en œuvre permet d'interroger l'état du variateur, de visualiser les variables, les paramètres, les entrées et les sorties et modifier les variables.

En mode multi drive, sélectionné le variateur à interroger



↳ La fenêtre Terminal est composé de 3 zones :

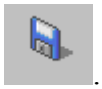
Zone d'état des tâches : permet de visualiser l'état des tâches du variateur, et la ligne en cours d'exécution.

Zone de visualisations : permet de visualiser le contenu d'une variable, d'un paramètre, d'une entrée ou d'une sortie.

Pour ajouter une variable ou un paramètre, cliquer sur l'icône et double cliquer sur une variable ou un des paramètres de cette fenêtre, le nom associé apparaît alors dans la zone de visualisation du terminal.

Pour supprimer une variable ou un paramètre, sélectionner le dans la zone de visualisation du terminal, cliquer sur l'icône .

Le nombre de variables ou paramètres à visualiser est limité à 16.

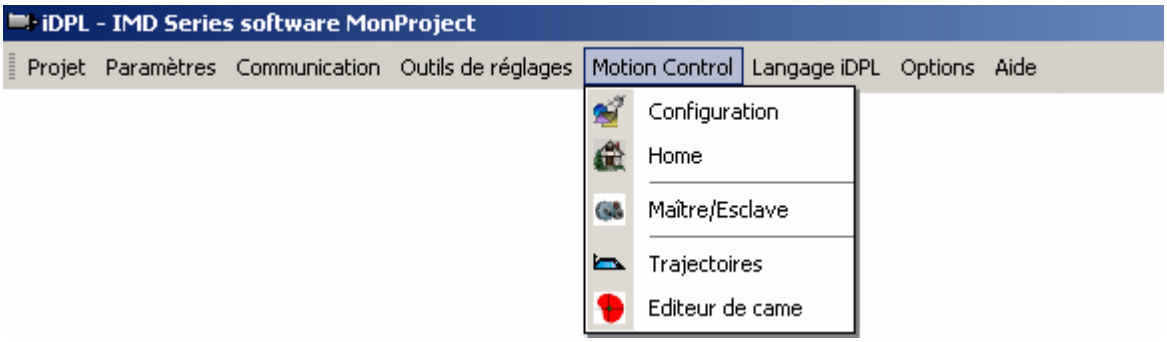
Il est possible d'ouvrir ou de sauver la configuration de l'hyper terminal avec les icônes : ,  .

Zone de saisie : permet de modifier le contenu d'une variable grâce à une zone de saisie blanche et de visualiser les modifications effectuées grâce à une zone de visualisation noire.

Il est possible de lire ou écrire les variables de type : VF, VB, VI, VL et VR mais aussi les variables de la FRAM : FI (entier), FL (entier long) et FR (réel) ; pour les variables entier long et réel de la FRAM, cela correspond aux 2 adresses consécutives en FRAM.

3-3-5- Motion control

Menu disponible seulement en mode position

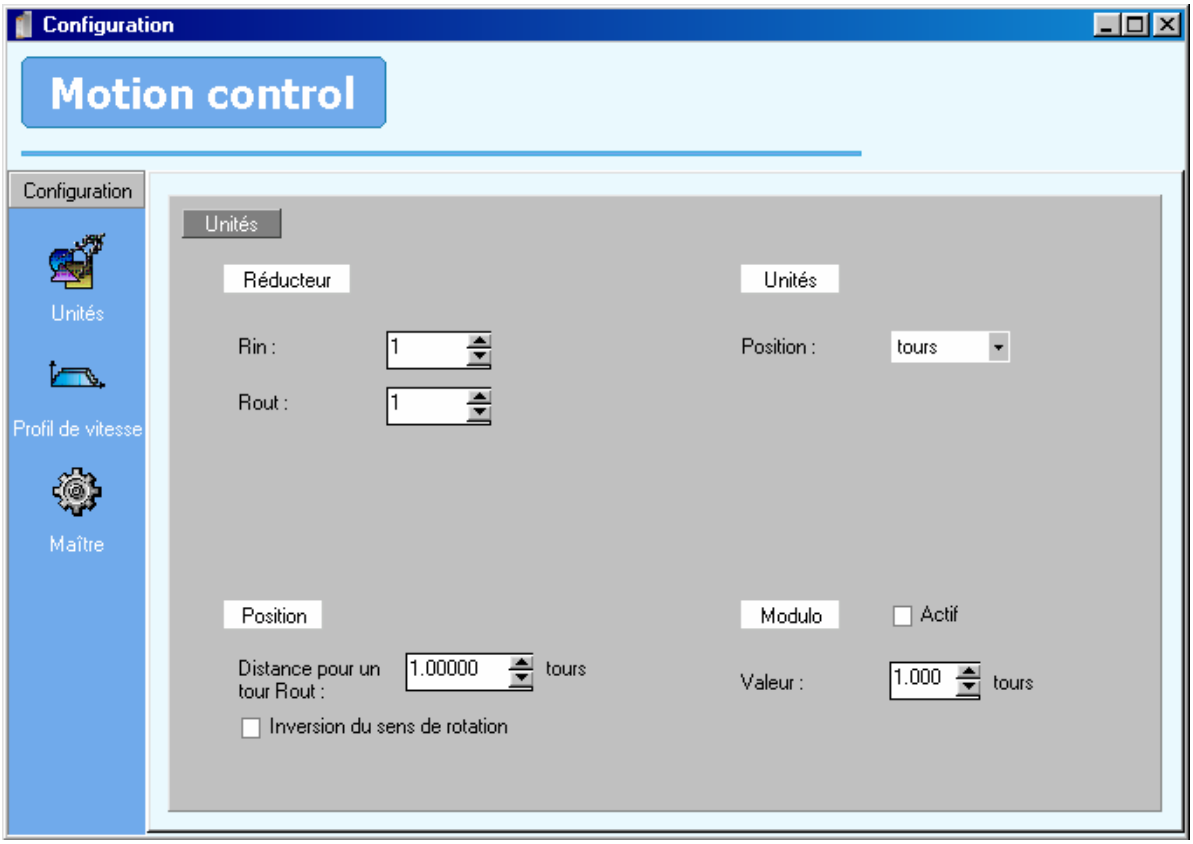


A) Configuration :

Icône :

Action : Permet de rentrer l'unité de travail (mm, degré …) ainsi que les vitesses, accélération et décélération par défaut.

- Les unités :



Exemple 1 : Axe infini

Moteur en bout de vis à bille au pas de 5mm. Unités = mm, Rin = 1, Rout = 1, Distance par tour = 5.000, Modulo non activé

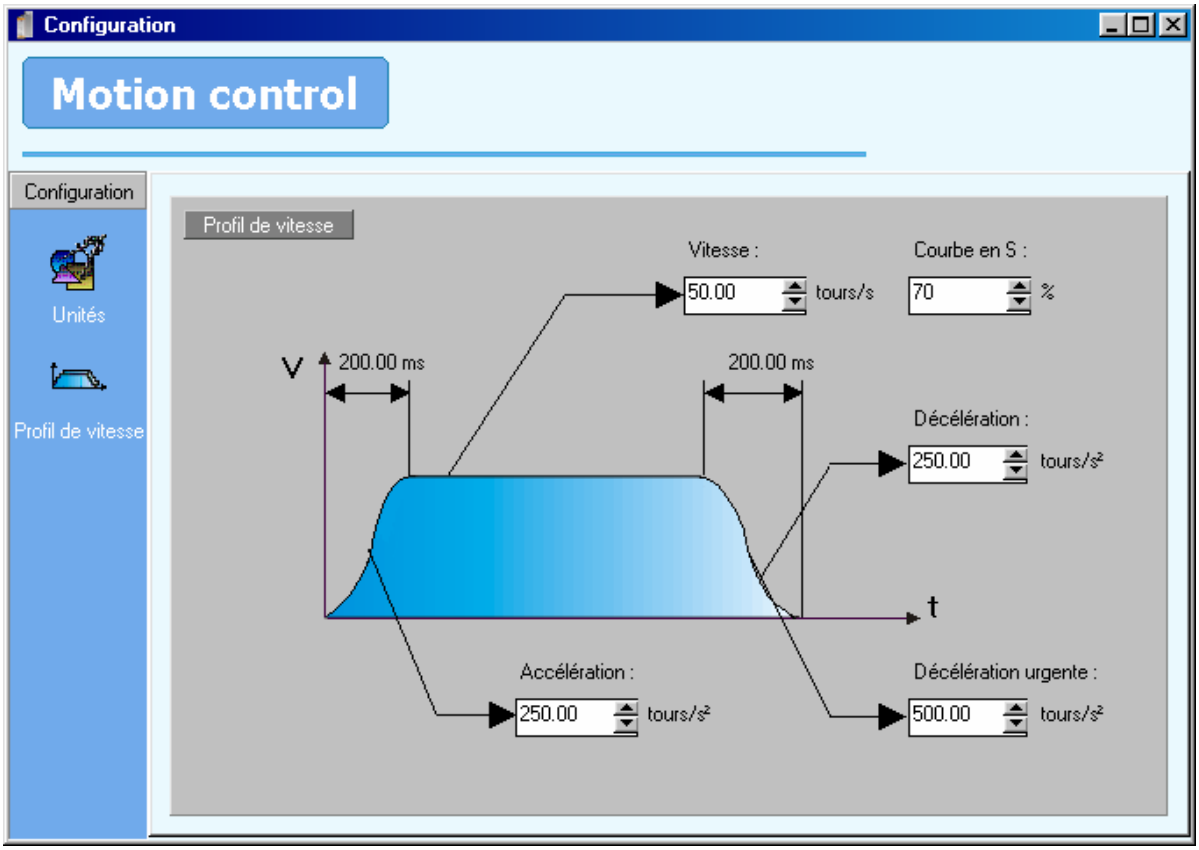
Exemple 2 : Axe infini

Moteur avec réducteur de 10. En sortie de réducteur, tourelle 360°, Unités = degrés, Rin = 10, Rout = 1, Distance par tour = 360.000, modulo activé avec une valeur de 360.000

Nota : le nombre de chiffres après la virgule est paramétrable dans le menu *Options / Langage iDPL*

Attention : Cet écran défini le ratio entre le retour de position et la sortie mécanique (en double boucle, le ratio se fait entre l'entrée de la boucle de position et la mécanique)

B) Le profil de vitesse :



Les vitesses, accélérations, décélérations exprimées en pourcentage dans le générateur, dans les trajectoires, dans les instructions ACC%, DEC%, et VEL% du langage iDPL font référence à ses valeurs

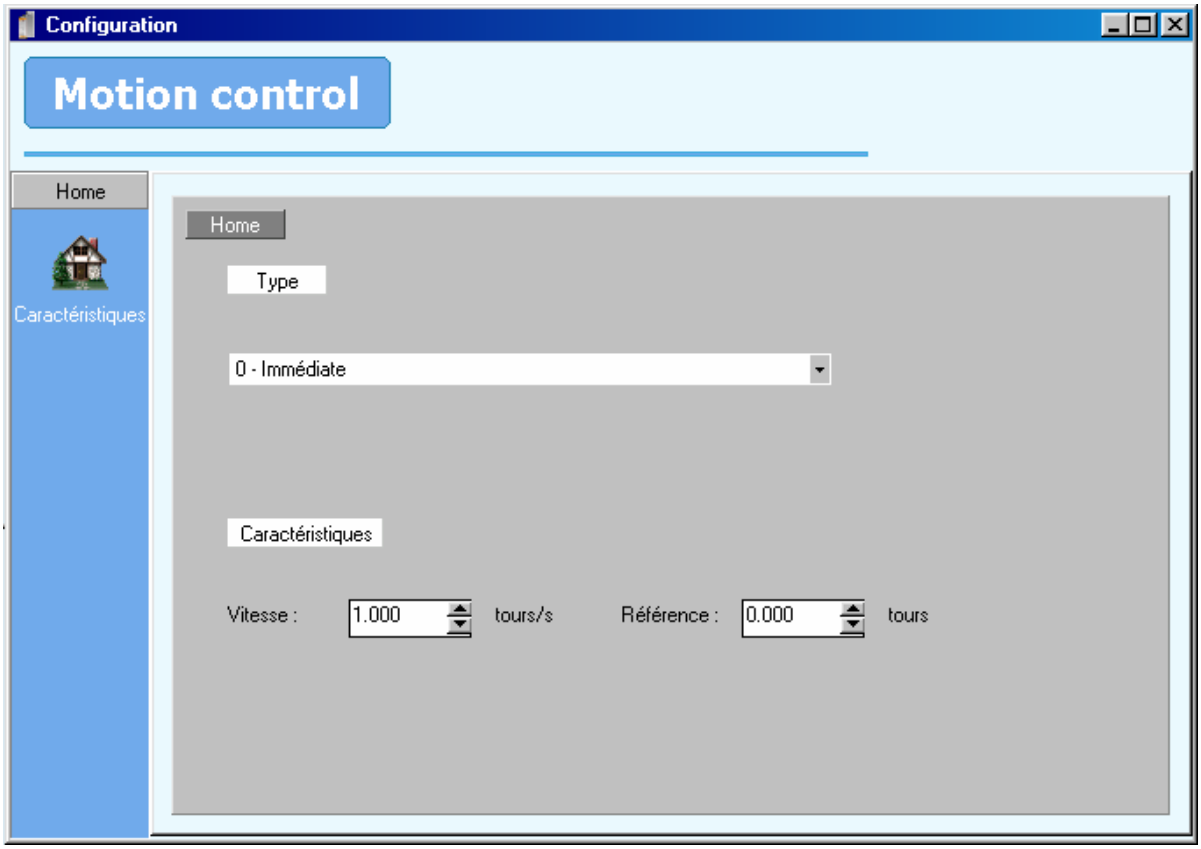
La décélération urgente est utilisée pour arrêter le mouvement si les entrées E2, E3 sont paramétrées en Fin de course.

Le paramètre Cœfficient S permet d'avoir des accélérations et décélérations en forme de S, ce qui permet d'adoucir les débuts et fins de mouvement. L'accélération avec un coefficient S peut varier de 0 à 200% de l'accélération donnée dans le profil de vitesse.

C) Home :

Icône :

Action : Permet de configurer la fonction de prise d'origine de l'axe.

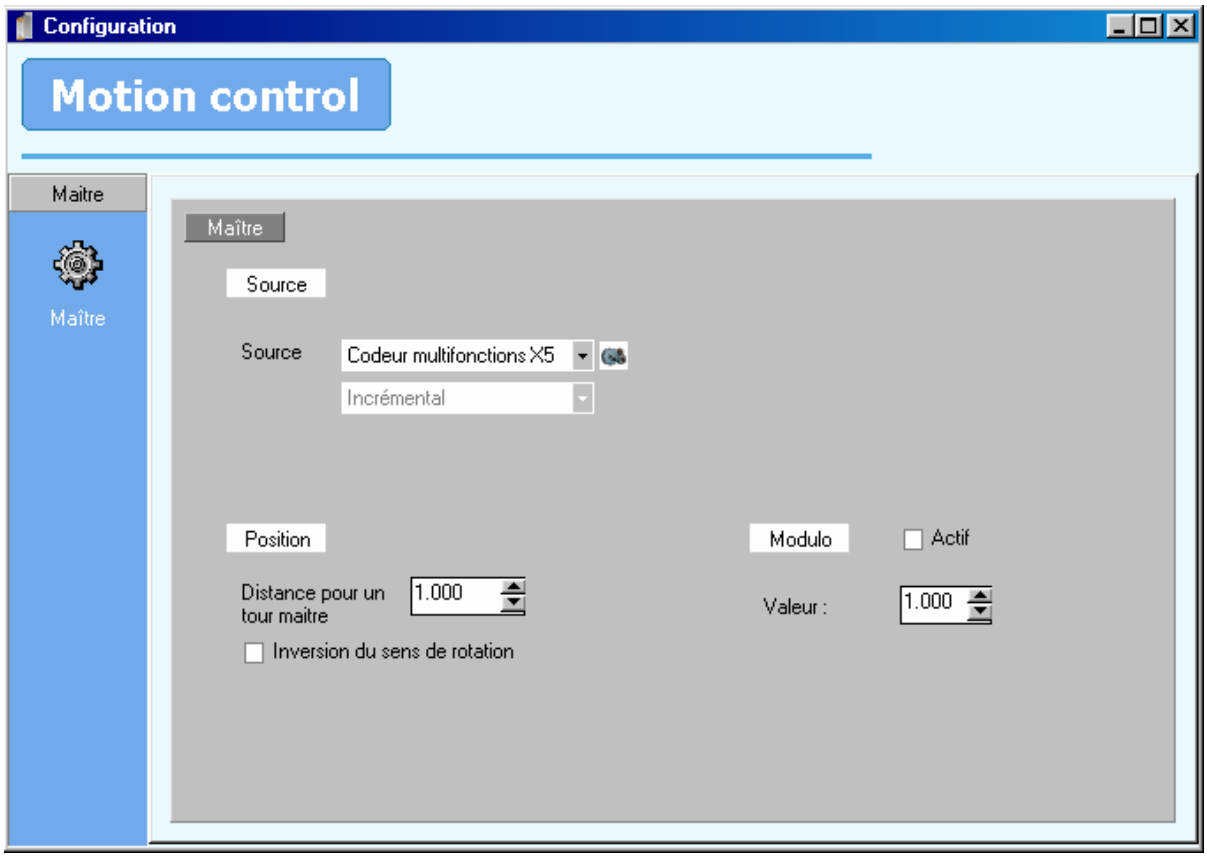


- Saisir le type (voir « notice sur POM.pdf » sur le CD iDPL)
- Saisir la vitesse à laquelle sera effectué le cycle d'origine.
- Saisir la position à charger dans le compteur lors de la détection de l'origine (par défaut 0)

D) Maître/Esclave :

Icône :

Action : Permet de configurer la fonction codeur maître.



Le codeur maître utilise les mêmes unités que celle du l'axe moteur.

E) Trajectoires :

Action : Permet de lancer des trajectoires via les entrées du variateur.

Voir chapitre sur les trajectoires pré-enregistrées.

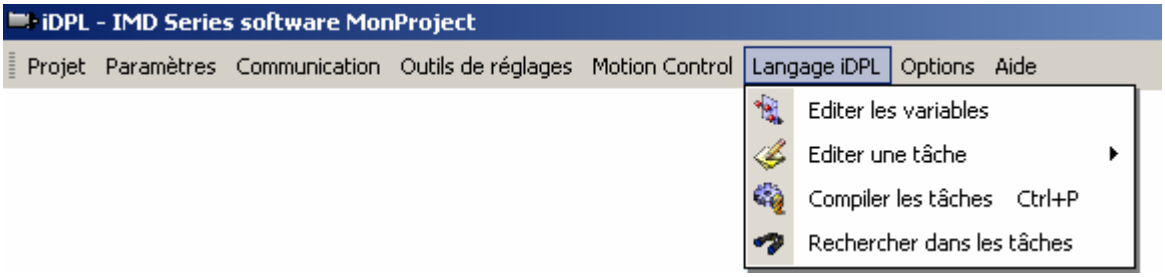
F) Editeur de came :

Icône :

Action : Permet d'éditer les cames.

Voir chapitre sur les cames.

3-3-6- Langage iDPL



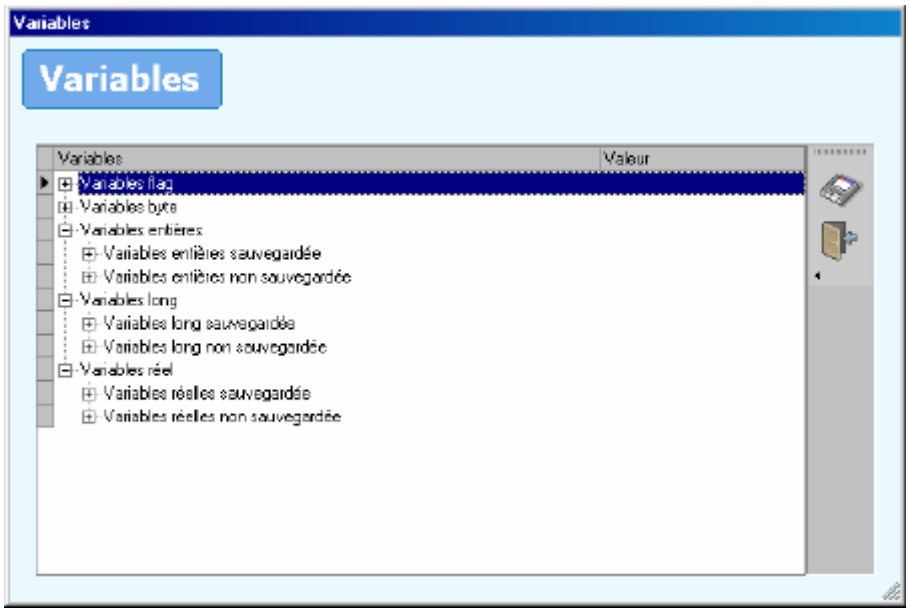
A) Editer les variables :

Icône :

Action : Permet de visualiser et modifier les variables du variateur (contenu dans le fichier dpv du répertoire variateur)

Pour mettre à jour ce fichier, lancer un réception des variables dans *Communication / Variables iDPL*

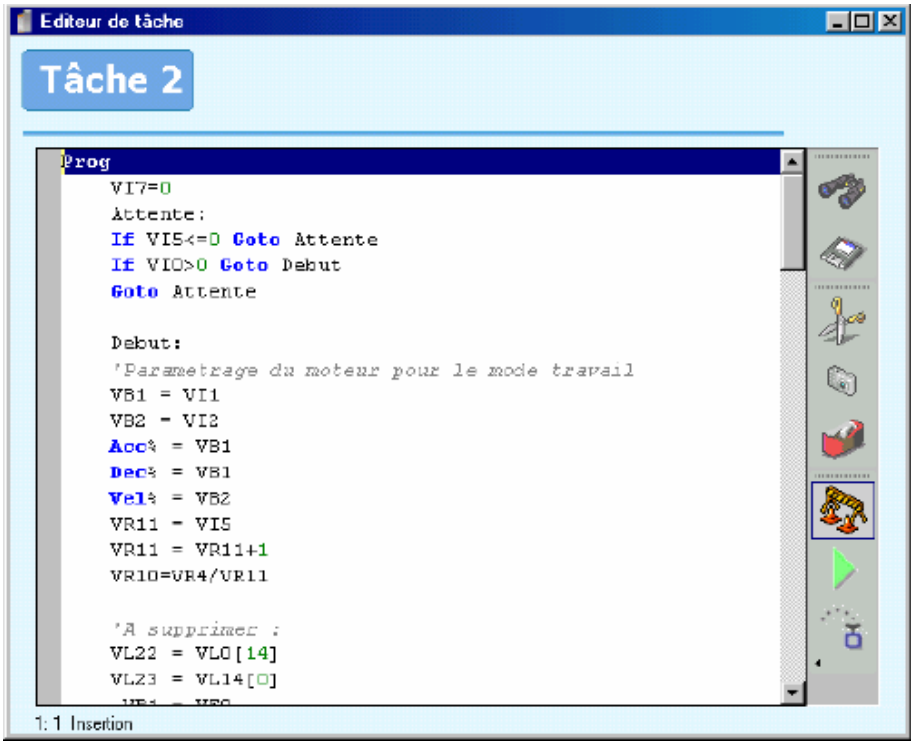
Pour mettre envoyer vos modifications, lancer un envoie des variables dans *Communication / Variables iDPL*



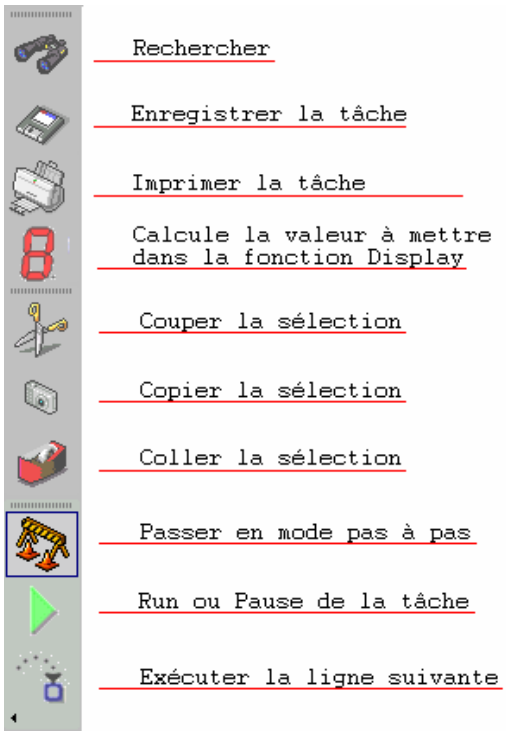
B) Editer une tâche :

Icône :

Action : L'éditeur de tâche se décompose en une zone d'édition de texte dans laquelle l'utilisateur vient entrer le code basic associé à son programme, une barre d'outils permettant l'aide à l'édition du code



Les outils de l'éditeur permettent de simplifier la programmation :


Rechercher
Enregistrer la tâche
Imprimer la tâche
Calcule la valeur à mettre dans la fonction Display
Couper la sélection
Copier la sélection
Coller la sélection
Passer en mode pas à pas
Run ou Pause de la tâche
Exécuter la ligne suivante

C) Compiler les tâches :

Icône :

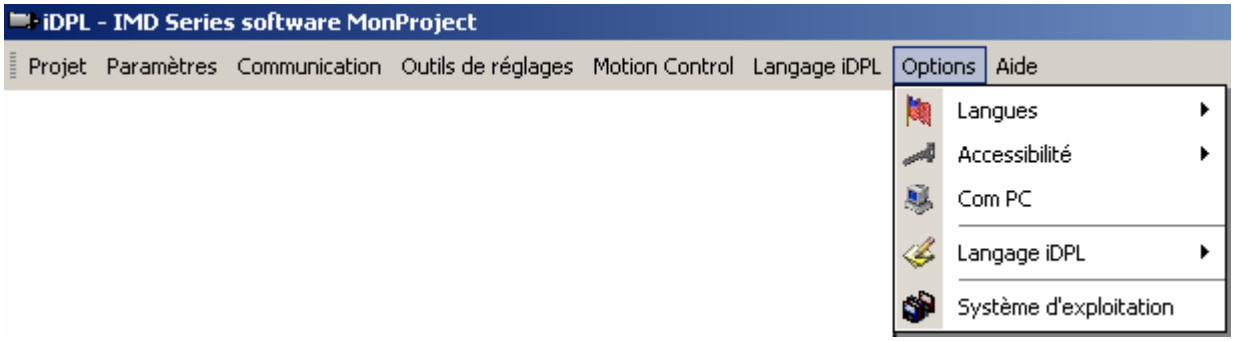
Action : Permet de compiler les tâches, pour vérifier la syntaxe du programme et créer le fichier binaire.

D) Rechercher dans les tâches :

Icône :

Action : Permet de rechercher une chaîne de caractère dans les tâches.

3-3-7- Options



A) Langues :

Icône :

Action : Ce sous-menu permet de choisir la langue dans laquelle le logiciel iDPL sera exploité.

B) Accessibilité :

Icône :

Action : Autorise l'accès aux différents niveaux de paramètres :

- Paramètres standards

- Paramètres avancés
- Paramètres usine

Et permet de cacher ou rendre visible le menu iDPL.

La modification des paramètres avancés peut entraîner la détérioration du variateur. Son accès est réservé à un personnel qualifié.

C) Com PC :

Icône :

Action : Sélection du port de communication du PC : com1, com2, com3 ou com4.

L'option *Communication système* permet de forcer la communication du PC et du variateur à un format figé : 57600 bauds, 8 bits de data, 1 bit de stop, pas de parité, adresse esclave = 1.

En *communication système*, les paramètres saisis dans le menu Paramètres / Liaison RS232 de base, ne sont pas gérés.

En activant *Communication système*, le PC utilise le signal RTS et le force au niveau logique 1. Dès que le variateur lit ce signal sur son entrée CTS, son format de liaison est forcé.

D) Langage iDPL :

Icône :

Action : Accès aux options du langage de programmation iDPL.

- Précision : définit le nombre de chiffres après la virgule pour tout ce qui est du type réel : les variables (VR0 à VR63), la position (POS_S dans le iDPL) …
- Temps de vieillissement : définit le temps maximal passé dans une tâche avant de basculer vers la suivante. Il est nécessaire de recompiler les tâches après une modification.

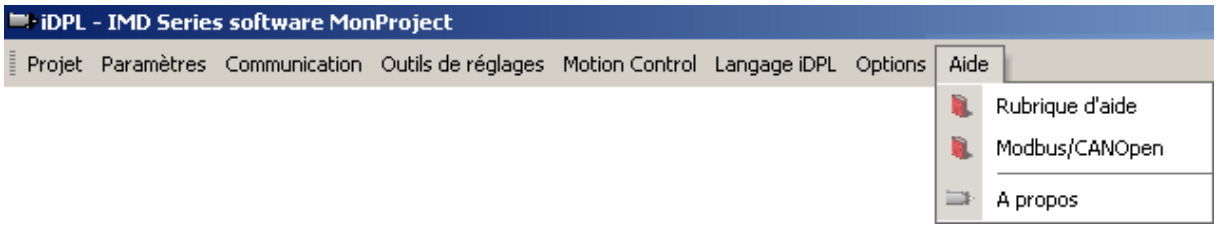
E) Système d'exploitation :

Icône :

Action : Chargement d'une nouvelle version d'operating system (firmware).

Réservé à un personnel qualifié : le changement de firmware efface les paramètres du variateur. Il est ensuite nécessaire de recharger le fichier de paramètres dans le variateur.

3-3-8- Aide



A) Rubrique d'aide :

Icône :

Action : Accès à la documentation complète.

B) A propos :

Icône :

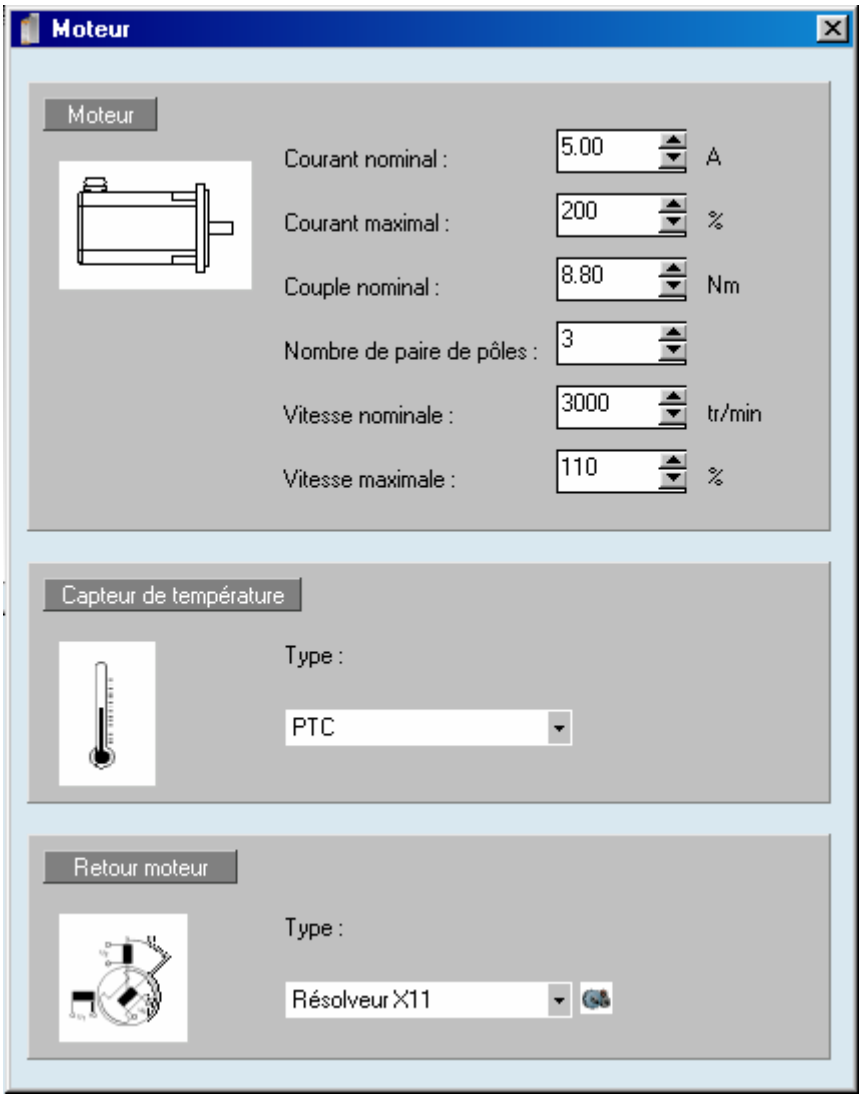
Action : Cette commande ouvre une boîte de dialogue indiquant la version du logiciel PC, la version du firmware, sa date de création, etc…

4- Réglage du variateur

4-1- Réglage des paramètres moteur et retour position

Si vous avez transféré le fichier de paramétrage correspondant au moteur à partir de la bibliothèque, vous n'avez pas de réglage moteur/retour moteur et de boucles d'asservissement à effectuer.

- Choix de la tension nominale du variateur dans les paramètres variateurs de la fenêtre paramètres. Sélectionner 230V ou 400V, cela modifiera automatiquement les paramètres de sécurités tel que sous tension ou résistance de freinage par rapport à votre tension d'alimentation.
- Sinon sélectionner le menu « **Paramètres/Moteur** ». Le menu suivant s'affiche :



A) Réglage moteur :

Se référer aux données constructeur ou à la plaque signalétique du moteur.

- Entrer les valeurs du moteur (courant nominal, vitesse nominal …).

Pour un usage normal, on mettra un courant maximal égale à 250% du courant nominal.

B) Réglage retour position :

- Sélectionner le type de retour position : résolveur ou SinCos.

a) Résolveur :

Le résolveur doit être du type TAMAGAWA TS2620N21E11 ou compatible. Pour tout autre type de résolveur, contacter notre service technique.

Vérifier grâce à l'oscilloscope que les signaux SINUS et COSINUS de votre résolveur évolue entre +0.9 et –0.9 :

1. Alimenter le variateur en 24V seulement (connecteur X6), le résolveur étant raccordé ainsi que la liaison RS 232.
2. Ouvrir le **tableau de bord** dans **outils de réglage**
3. Vérifier que la position évolue correctement.
4. Ouvrir l'**oscilloscope** dans **outils de réglage**.
5. Sélectionner les signaux SINUS et COSINUS dans RESOLVEUR puis lancer l'acquisition
6. Faite tourner le moteur à la main et visualiser les courbes obtenues. Si les signaux dépassent +0.9 ou –0.9, aller dans la liste des paramètres résolveur (accessible \ paramètres avancés) et baisser la valeur de *Gain excitation*. Si les signaux sont très faible (entre +0.5 et –0.5), contacter notre service technique.
7. Faire l'auto tuning.

b) SinCos :

1. Saisir la résolution du codeur et le type de lien série.
2. Ouvrir le **tableau de bord** dans **outils de réglage**
3. Vérifier que la position évolue correctement.
4. Faire l'auto tuning.

c) Auto tuning retour moteur :

- Réglage de l'offset résolveur/SinCos :

  1. Alimenter également la puissance sur le variateur.
  2. Aller dans **options** puis **accessibilité** et valider **paramètres avancés.**
  3. Aller dans **outils de réglage** et cliquer sur **auto tuning \ résolveur/SinCos.**

  Vérifier que le frein externe soit forcé.

  Le variateur asservit le moteur et mesure automatiquement l'offset résolveur, cette étape ne dure environ 5 secondes et est indiqué par une barre de progression.

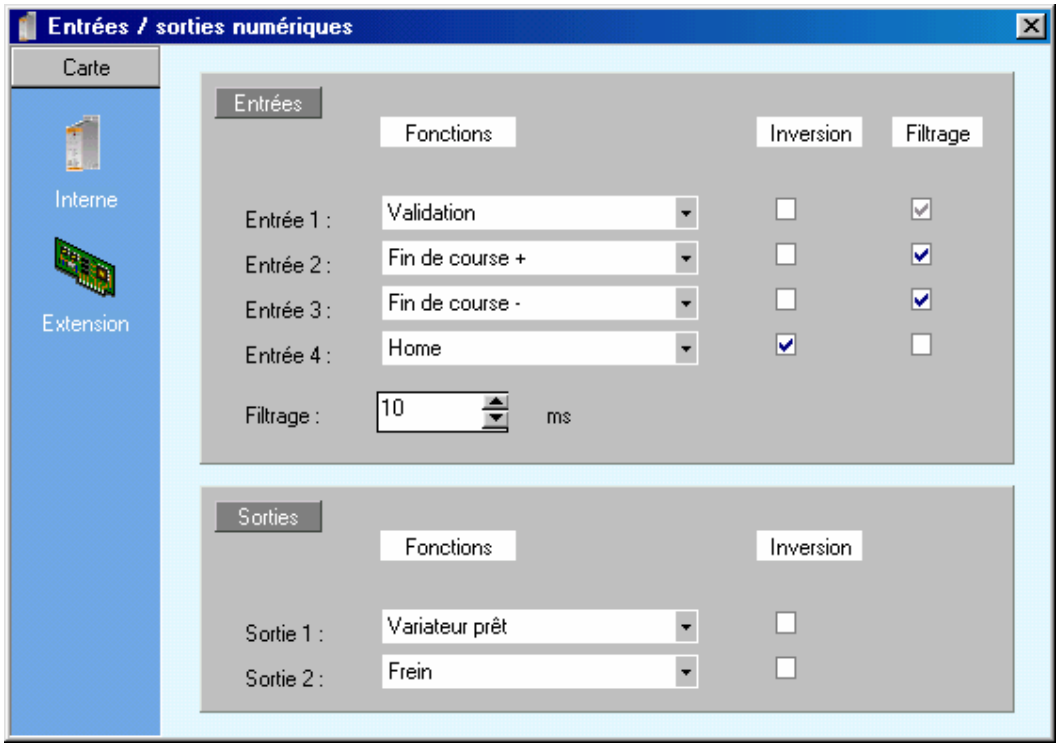
- Fermer la fenêtre de paramétrage.
- Sauvegarder les paramètres.

Nota : Si des phases moteurs sont inversées, l'auto tuning l'indiquera.

4-2- Réglage du mode de déverrouillage variateur

Pour déverrouiller le variateur, on doit sélectionner l'entrée de verrouillage. Celle-ci décide quelles conditions sont requises.

- Sélectionner le menu **Paramètres/Entrées sorties TOR.**



- Sélectionner **Aucune** dans le champ **Entrée E1**. (A la fin des réglages des boucles de régulation, penser à remodifier le fonction de l'entrée E1 selon vos besoins).

  Le bouton Enable de l'écran principal permet alors de déverrouiller ou non le variateur.

- Sauvegarder les paramètres.

4-3- Les modes de fonctionnement

Le variateur iMD gère 3 modes de fonctionnements utilisant différentes boucles de régulation.

- **MODE COUPLE** Boucle de courant

En mode couple, le moteur maintient le couple spécifié. La vitesse dépend de la charge appliquée.

- **MODE VITESSE** Boucle de courant

  Boucle de vitesse

En mode vitesse, le moteur maintient la vitesse spécifiée quelle que soit la charge.

- **MODE POSITION** Boucle de courant

  Boucle de vitesse

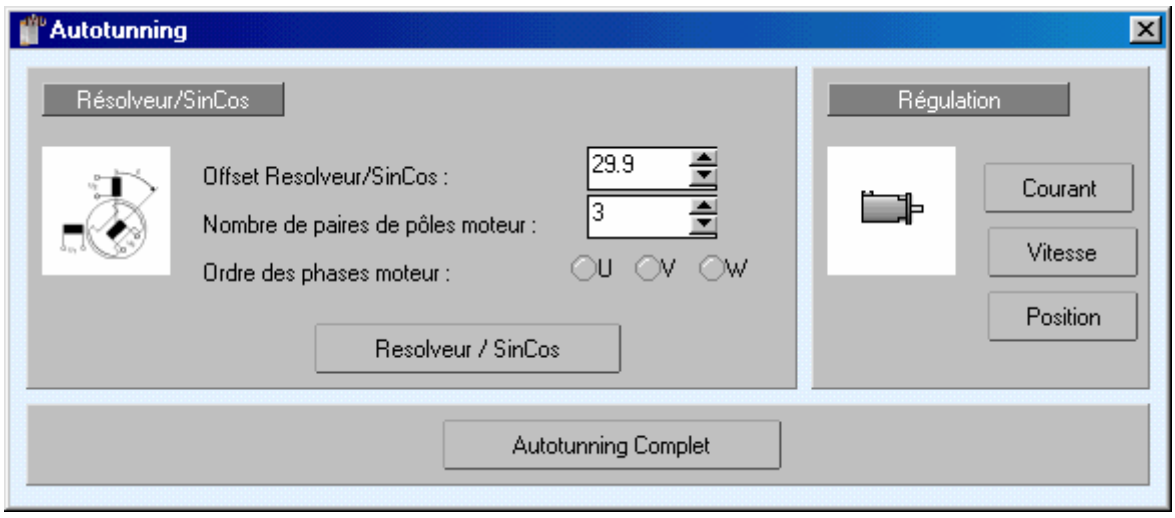
  Boucle de position

En mode position, le moteur suit un profil de trajectoire demandée.

Le choix du mode de fonctionnement se fait à partir de la fenêtre PARAMETRES à la ligne variateur. Sélectionner l'un des trois modes (COUPLE, VITESSE, POSITION)

Le variateur est automatiquement déverrouillé lors d'un changement de mode.

4-4- Réglage automatique des boucles de régulation



4-4-1- Auto tuning de la boucle de courant :

Durant cette phase le moteur va effectuer des mouvements très petits pour calculer la limite de vibrations puis des mouvements d'amplitude plus élevés (en fonctions de l'inertie)

Attention : il est possible de faire cette phase d'auto tuning moteur à vide ou accouplé mais il est recommandé de faire cette dernière à vide si la mécanique est fragile.

4-4-2- Auto tuning de la boucle de vitesse :

Durant cette phase le moteur va effectuer plusieurs tours à vitesse moyenne.

Attention : L'axe doit être de type infini car le nombre de tour moteur nécessaire pour l'auto tuning n'est pas connu. Pour un réglage optimum, il est nécessaire de raccorder la mécanique sinon l'axe manquera de raideur.

4-4-3- Auto tuning de la boucle de position :

Durant cette phase le moteur va effectuer plusieurs tours à vitesse faible.

Attention : La mécanique peut être ou ne pas être raccordée.

4-4-4- Auto tuning complet :

Permet d'enchaîner les différents auto tuning : du retour position à la boucle de position.

4-4-5- Mise en garde sur l'auto tuning :

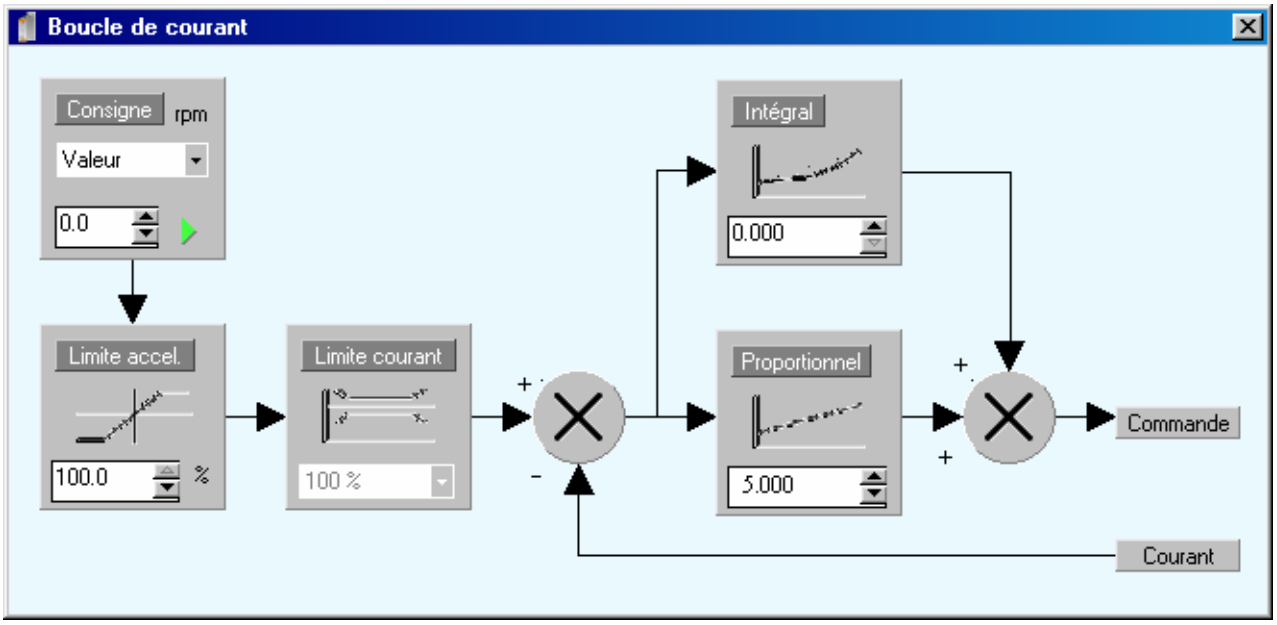
- Pendant l'auto tuning, toutes les sécurités sont actives (I^2t etc ...)
- Pour diminuer/annuler les « bruits » en début et fin de trajectoires, mettre à 0 la compensation d'accélération dans la boucle de vitesse (l'erreur de poursuite sera alors plus élevée durant les phases d'accélérations et décélérations
- Pour augmenter la raideur du système, augmenter le gain proportionnel de la boucle de vitesse.
- Pour augmenter le temps de réponse du système, augmenter le gain intégral de la boucle de vitesse.
- En cas d'instabilité du système diminuer/annuler le gain intégral de la boucle de vitesse.

4-5- Réglage manuel des boucles de régulation

4-5-1- Réglage de la boucle de courant

Le bon réglage de la boucle de courant est indispensable pour adapter la boucle de vitesse lors des étapes suivantes. Les paramètres sont le **gain intégral** et le **gain proportionnel**. Ce réglage est directement lié aux caractéristiques du moteur et ne dépend pas de la charge.

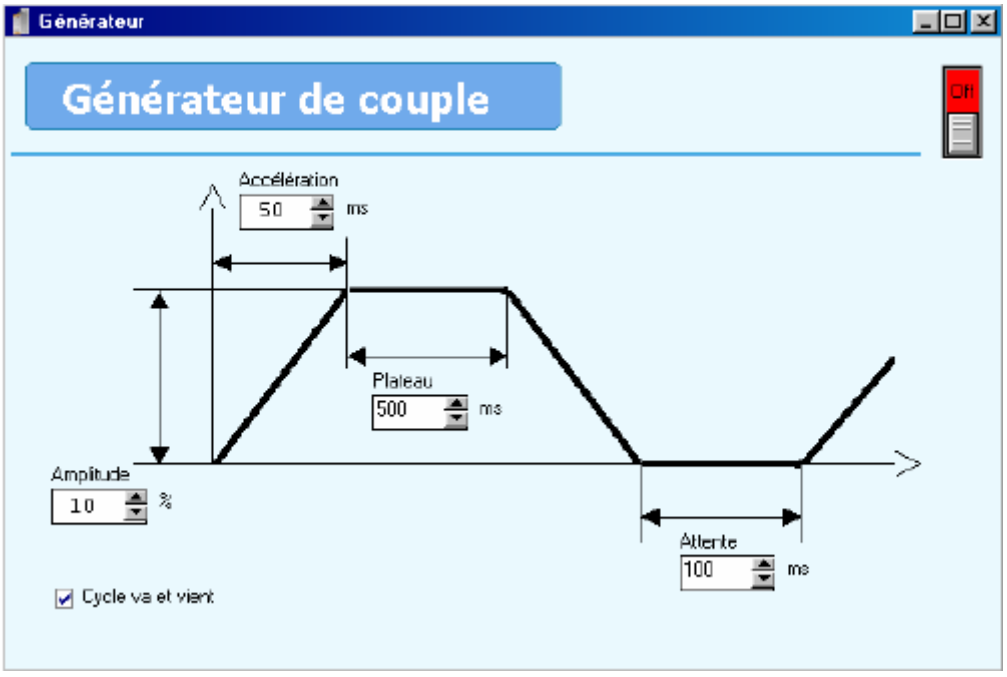
- Verrouiller le variateur (bouton *Enable* sur OFF dans l'écran principal).
- Sélectionner le variateur en mode couple à partir de la fenêtre principale.
- Sélectionner le menu **Paramètres / Boucle de courant**. Le menu suivant s'affiche :



Pour commencer le réglage de la boucle de courant, prendre les réglages ci dessus.

La consigne doit être du type **valeur**.

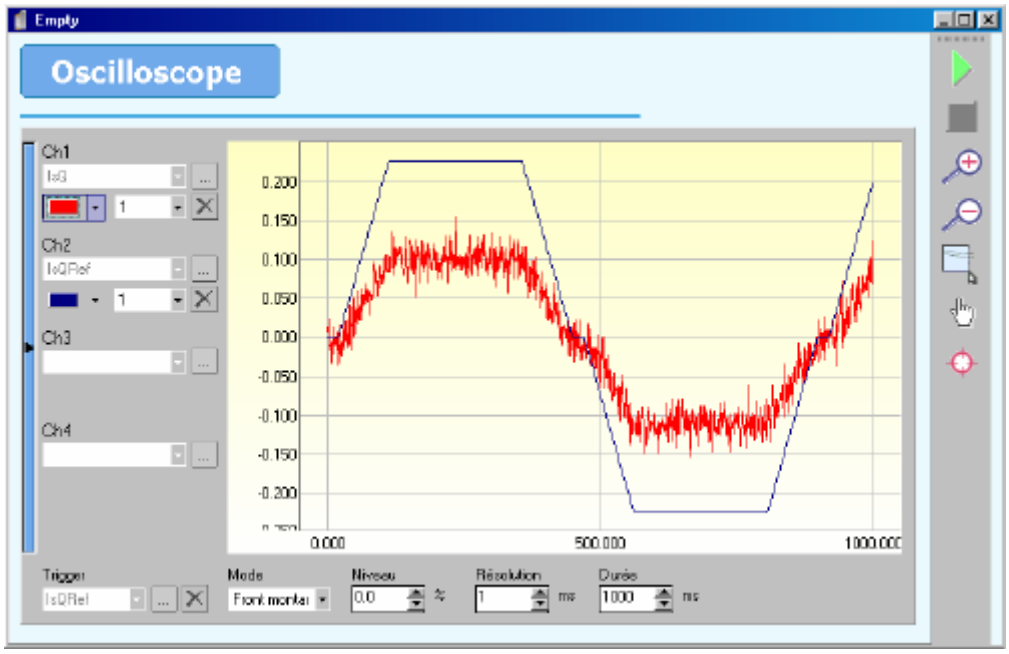
- Dans **Outils de réglages / Générateur**, lancer un mouvement comme ci dessous :



Vous pouvez faire varier l'amplitude de 5 à 15 % et l'accélération de 50 à 100%, selon le type de moteur. L'amplitude est exprimée en pourcentage du courant maximal du moteur.

Pour lancer un mouvement vous devez asservir le variateur par le bouton *Enable* en position ON sur l'écran principal.

- Aller dans **Outils de réglages / Oscilloscope** pour visualiser ce type de courbe du courant durant le mouvement :



1. Sélectionner **IsQ** dans **Boucle de courant** pour la voie 1.
2. Sélectionner **IsQREF** dans **Boucle de courant** pour la voie 2.
3. Sélectionner **IsQREF** pour le trigger et choisir front montant.

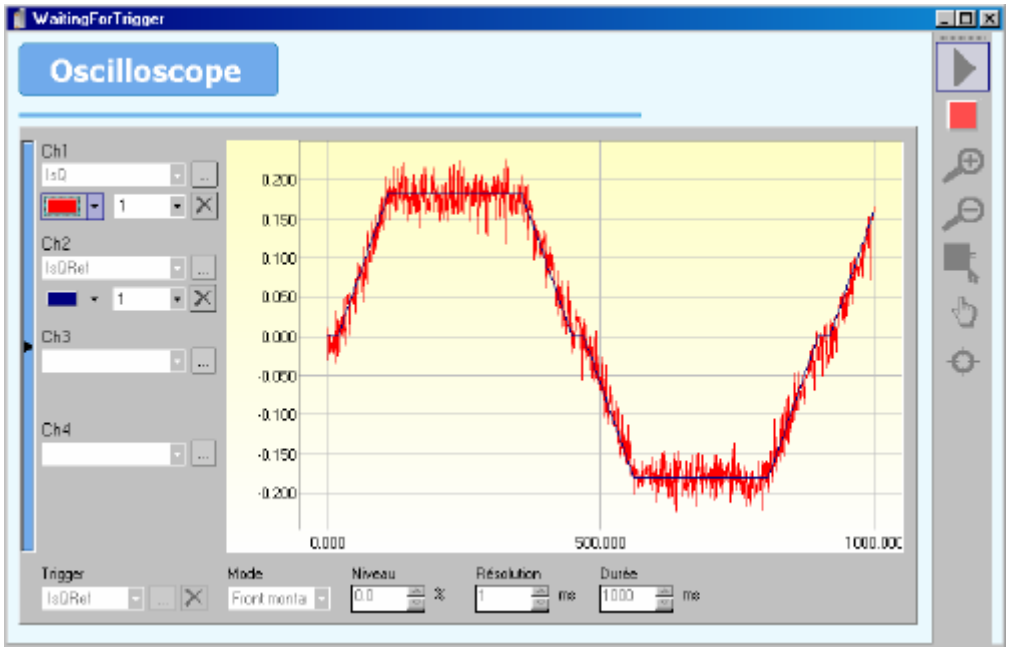
Si le signal IsQREF n'a pas la forme d'un trapèze modifier alors les valeurs *Amplitude* et *Accélération* dans la fenêtre oscilloscope.

- Avant de commencer, il est préférable de bloquer l'arbre du moteur (par exemple avec la main dans le cas de petits moteurs).

1. Augmenter le gain proportionnel jusqu'à ce que le courant réel (IsQ) s'approche le plus près possible de la consigne (IsQREF).
2. Si le moteur se met à vibrer, baisser le gain proportionnel de 20%.
3. Augmenter légèrement le gain intégral jusqu'à ce que le courant réel suive parfaitement la consigne.

Valeurs usuelles : gain proportionnel de 5 à 250, gain intégral de 0.5 à 10.

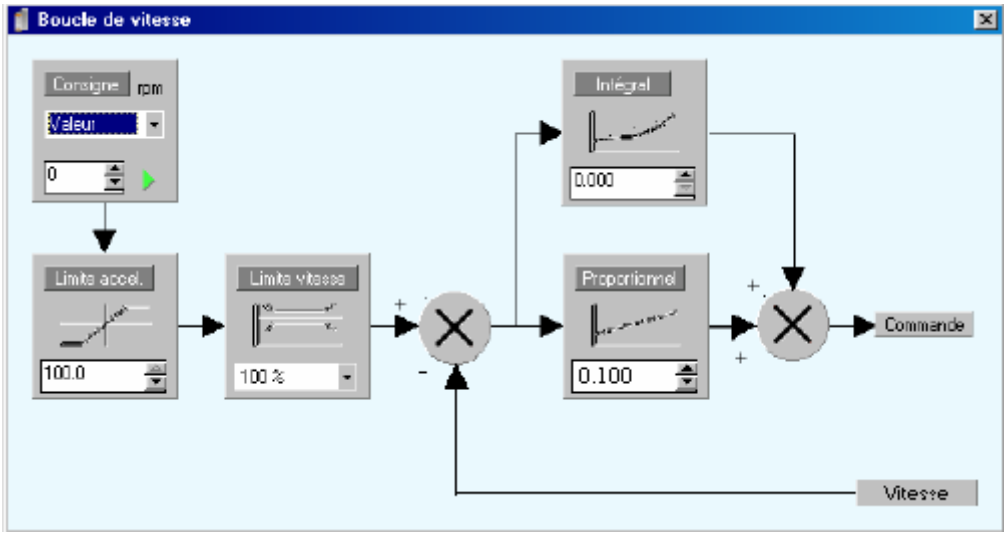
Exemple de courbes avec gain proportionnel et intégral optimisés



- Sauver les réglages avec **Paramètres/Sauvegarder les paramètres**.

4-5-2- Réglage de la boucle de vitesse

- Verrouiller le variateur (bouton *Enable* sur OFF dans l'écran principal).
- Sélectionner le variateur en mode **vitesse** à partir de la fenêtre principale.
- Sélectionner le menu **Paramètres / Boucle de vitesse**

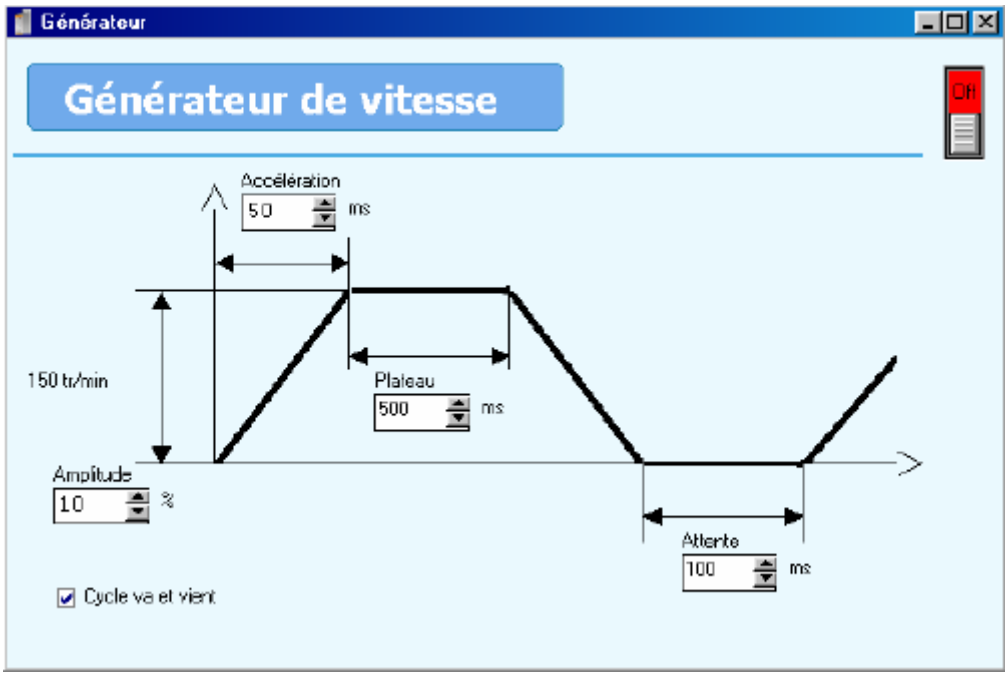


1. Pour commencer le réglage de la boucle de vitesse, prendre les réglages ci dessus.

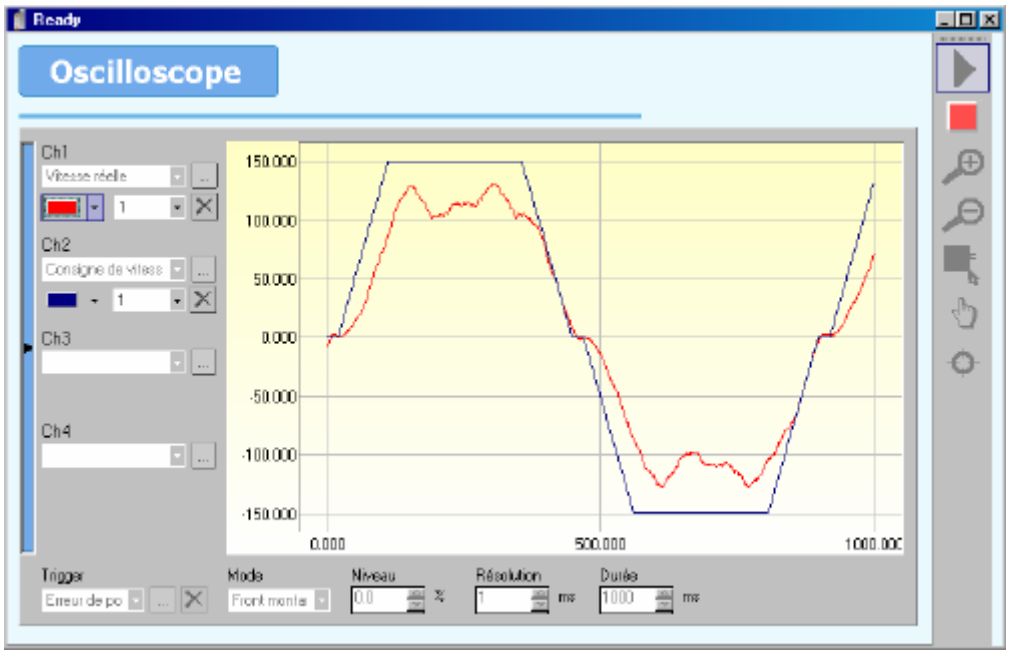
La consigne doit être du type **valeur**.

- Déverrouiller le variateur (bouton *Enable* sur ON dans l'écran principal).
- Dans **Outils de réglages / Générateur**, lancer un mouvement comme ci dessous :

L'arbre du moteur ne doit pas être bloqué. Un réglage optimal de la boucle de vitesse, s'effectue avec le moteur en charge.



- Aller dans **Outils de réglages / Oscilloscope** pour visualiser ce type de courbe de vitesse :



1. Sélectionner **Vitesse réelle** dans **Boucle de vitesse** pour la voie 1.
2. Sélectionner **Consigne de vitesse** dans **Boucle de vitesse** pour la voie 2.
3. Sélectionner **Consigne de vitesse** pour le trigger et choisir front montant.

Si le signal Consigne de vitesse n'a pas la forme d'un trapèze, modifier alors les valeurs d'**Amplitude** et d'**Accélération** dans la fenêtre générateur.

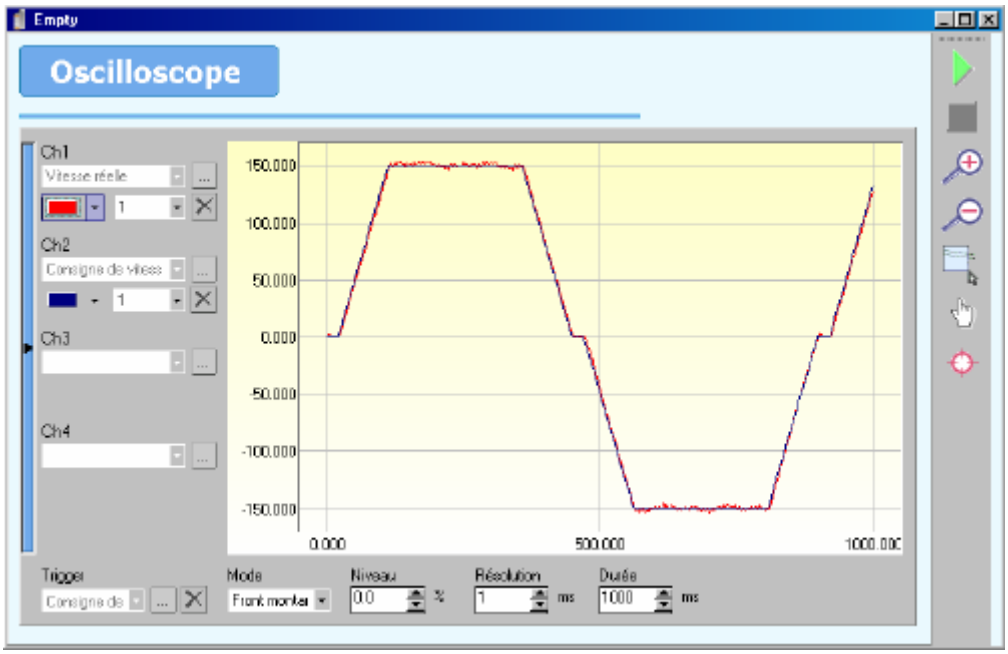
- Augmenter le **gain proportionnel** jusqu'à ce que la vitesse réelle s'approche le plus près possible de la consigne.

Si le moteur se met à vibrer, baisser le **gain proportionnel** de 20%.

Augmenter le **gain intégral** jusqu'à ce que le vitesse réelle suive parfaitement la consigne.

Valeurs usuelles : gain proportionnel de 100 à 5000, gain intégral de 2 à 200.

Exemple de courbes avec gain proportionnel et intégral optimisés

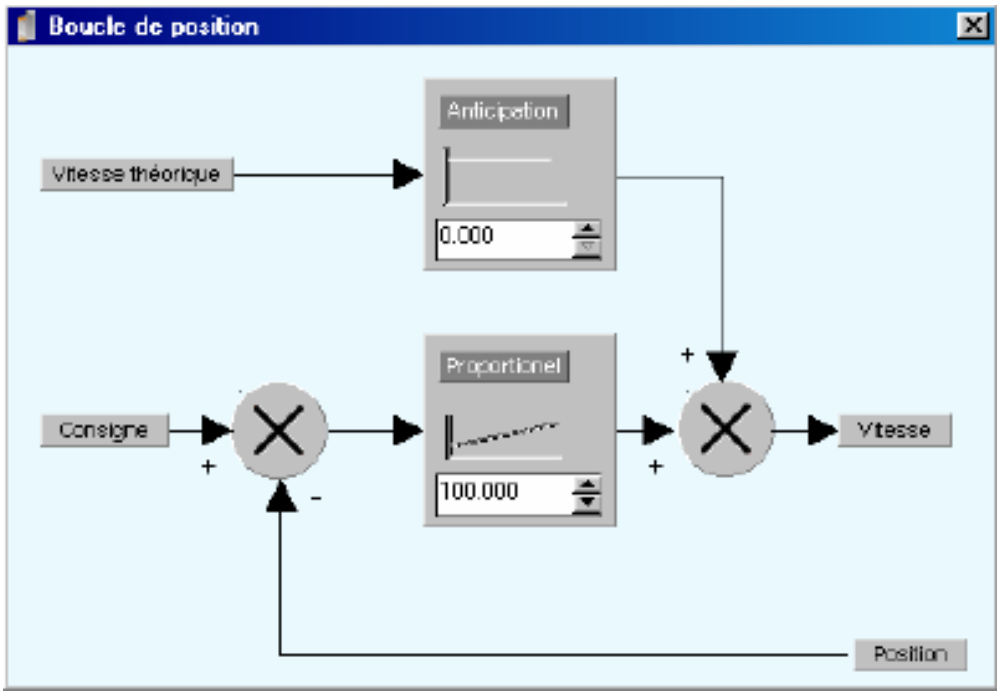


- Sauver les paramètres avec **Paramètres/Sauvegarder les paramètres.**

4-5-3- Réglage de la boucle de position

Le réglage de la boucle de position se fait en demandant des déplacements à partir de la fenêtre générateur.

- Verrouiller le variateur (bouton *Enable* sur OFF dans l'écran principal).
- Sélectionner le variateur en mode **Position** à partir de la fenêtre principale.
- Sélectionner le menu **Paramètres/Boucle de position**

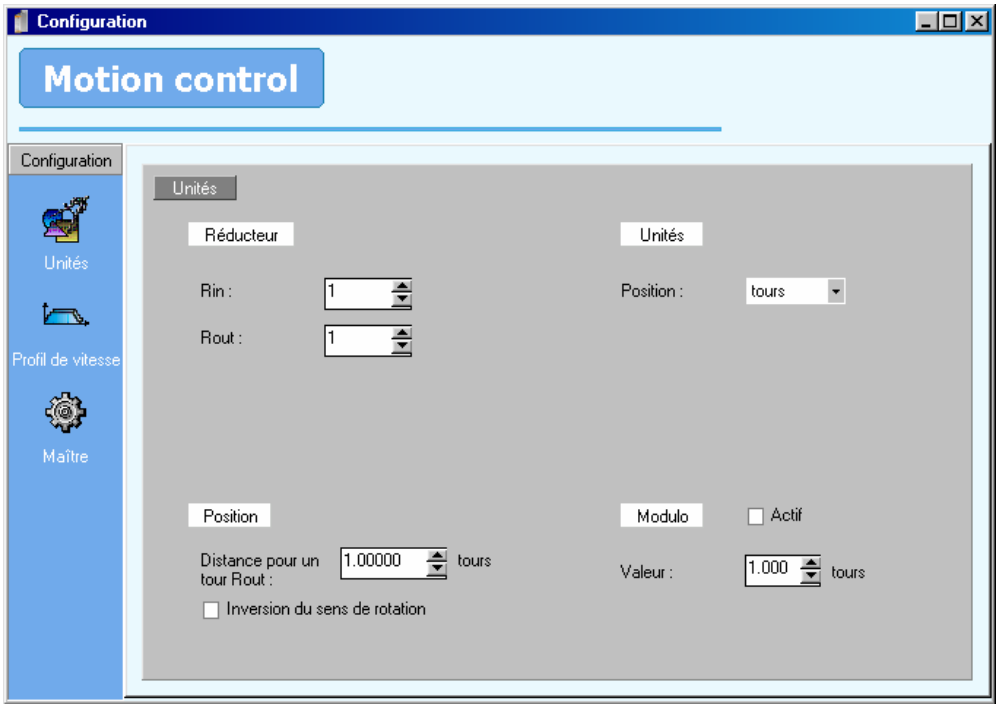


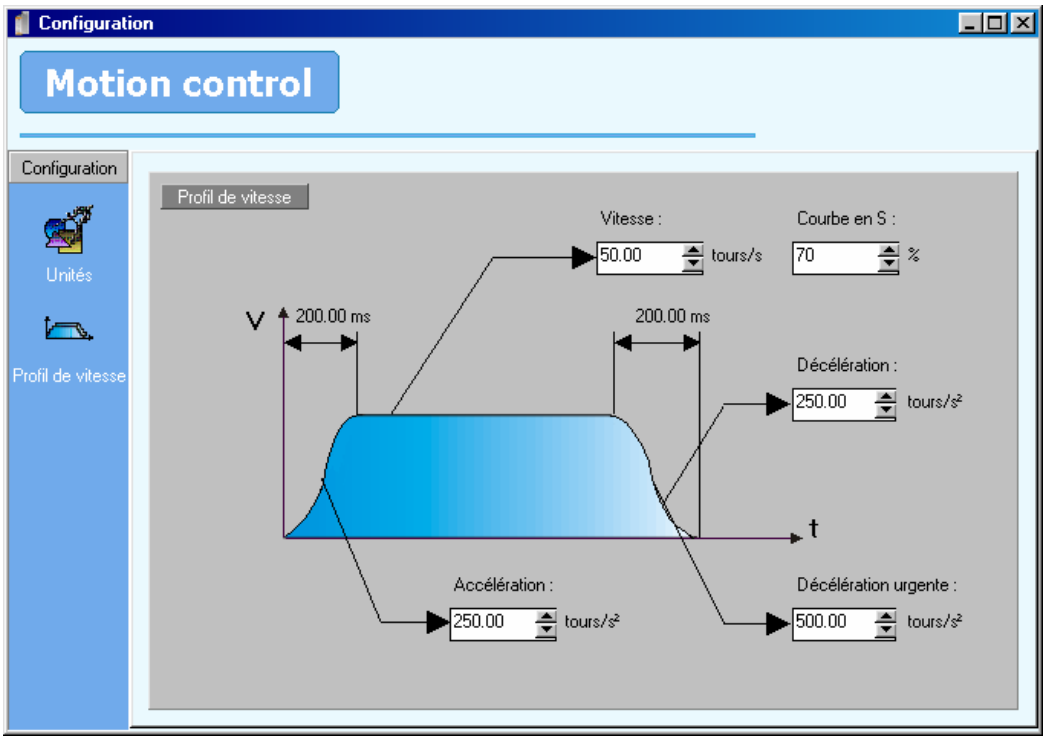
Pour commencer le réglage de la boucle de position, prendre les réglages ci dessus.

- Dans **Motion control** / **Configuration**, modifier les unités et le profil de vitesse pour correspondre à votre besoin :

  Exemple pour un moteur avec une vitesse nominale de 3000 tr/min

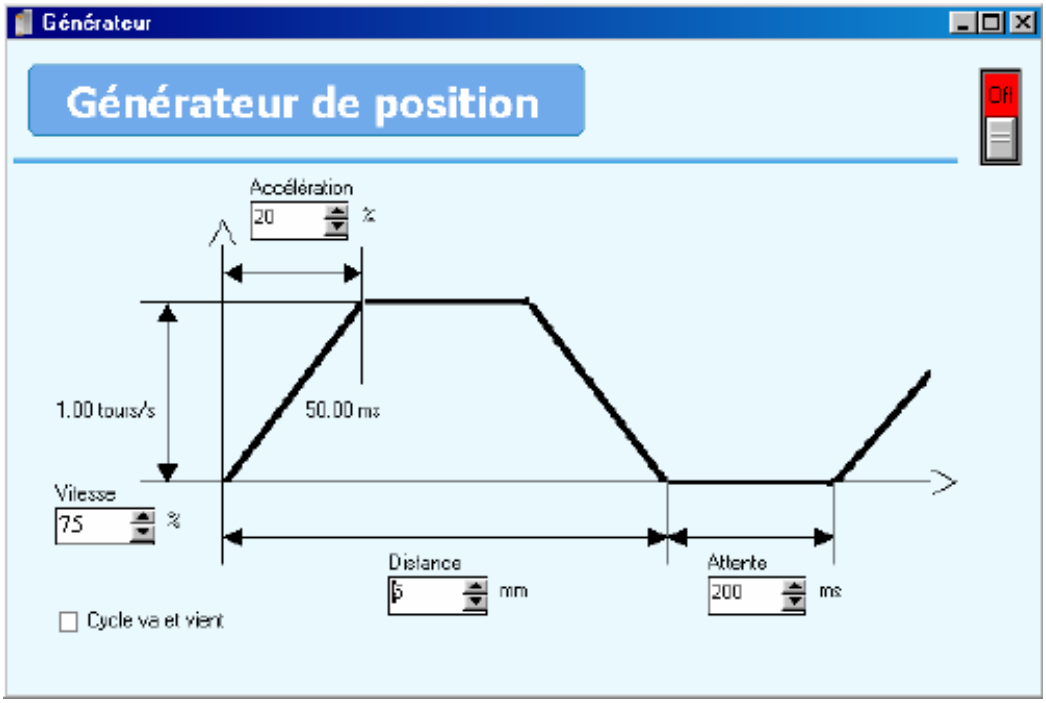
  Le pourcentage de vitesse et d'accélération que l'on rentre dans la fenêtre du générateur fait référence à la vitesse et à l'accélération données dans le menu **Motion control** / **Configuration** / **Profil de vitesse.**



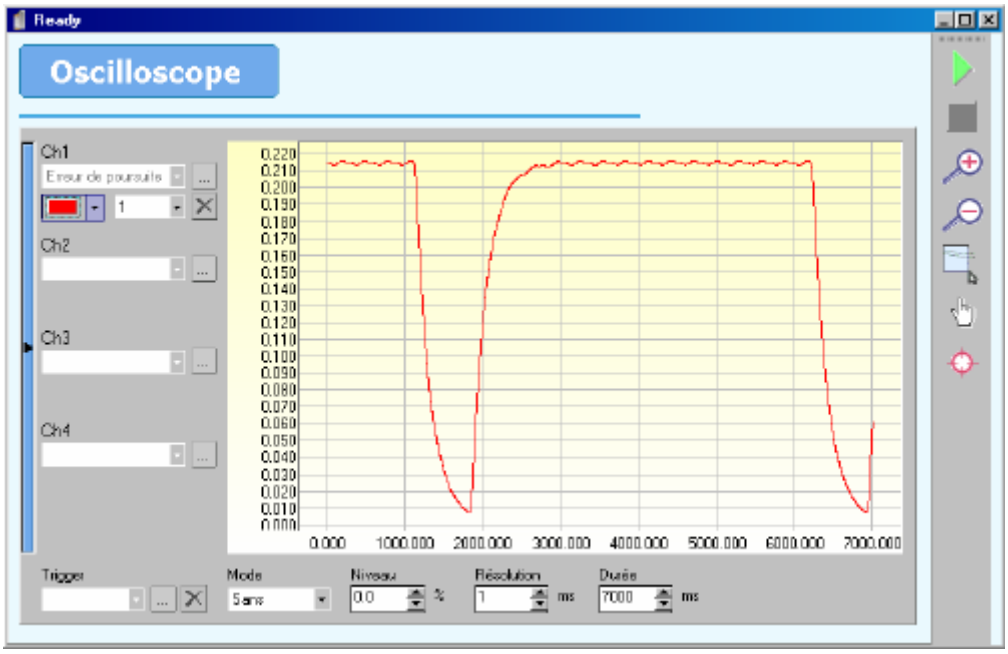


Selon les caractéristiques de votre moteur, pensez à régler votre erreur de poursuite dans **Paramètres** / **Sécurité** / **Position** / **Erreur de poursuite**

- Dans **Outils de réglages / Générateur**, lancer un mouvement comme ci dessous :



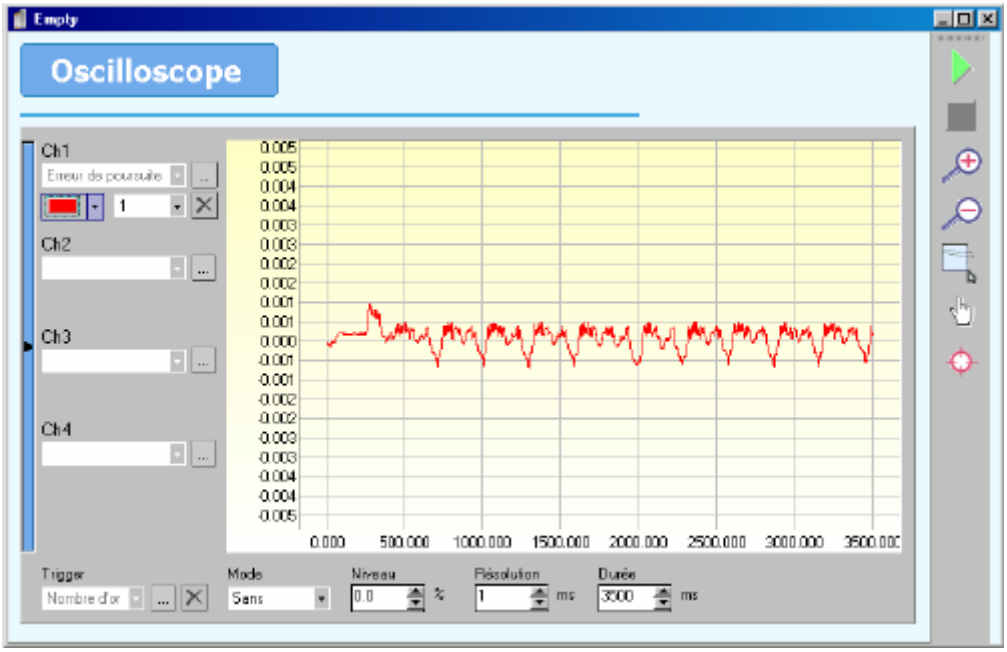
- Aller dans **Outils de réglages / Oscilloscope** pour visualiser ce type de courbe d'erreur de poursuite :



- Sélectionner **Erreur de poursuite** dans **Boucle de position** pour la voie 1.
- Ne pas sélectionner de fonction trigger.
- Augmenter le **gain proportionnel** jusqu'à atteindre la limite de la stabilité du moteur puis le baisser de 20%.
- Augmenter le **gain anticipation** de vitesse pour tendre vers une erreur de poursuite nulle.

Valeurs usuelles : gain proportionnel de 1000 à 4000, gain anticipation d'environ 33.

Exemple de courbes avec gain proportionnel et anticipation optimisés



Nota : Il est également intéressant de visualiser sur le canal n°2 de l'oscilloscope la vitesse théorique afin de connaître la valeur de l'erreur de poursuite pendant les phases d'accélération et de décélération. Dans ce cas régler le canal n°1 avec un facteur de 1000 et le canal n°2 avec un facteur de 0.001

- Sauver les paramètres avec **Paramètres/Sauvegarder les paramètres.**

4-6- Autres réglages

4-6-1- Réglage en boucle de vitesse

1. Choisir le mode vitesse
2. Dans boucle de vitesse \ Source consigne, choisir Entrée ana
3. Dans E/S analogique, vérifier que l'échelle de la voie 1 consigne soit à 100%
4. Dans paramètres moteurs mettre la vitesse moteur à la vitesse nominale du moteur et mettre vitesse max à 110%
5. Dans Sécurités \ DCbus, activer la sous tension DCbus
6. Vérifier dans sortie codeur multifonction que le mode soit Bypass codeur incrémental, saisir la résolution et la source du retour position

4-6-2- Réglage en double boucle résolveur/codeur

1. Choisir le mode position
2. Dans paramètres\ régulation \ boucle, choisir double puis configurer la source de la boucle de position

   Ex : si codeur incrémental : sélectionner résolveur X11 puis saisir sa résolution

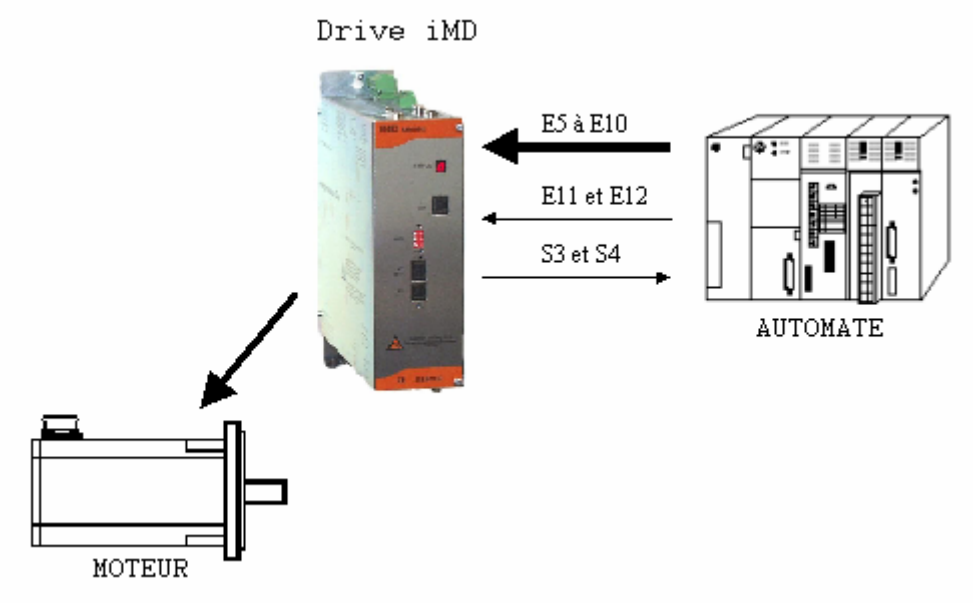
4-6-3- Réglage en entrée stepper

1. Choisir le mode position
2. Dans motion control / maître esclave, sélectionner entrée codeur multifonction comme source puis la configurer en mode stepper
3. Créer une tache avec les fonctions gearbox et startgearbox afin d'activer le liaison maître / esclave

5- Les trajectoires

5-1- Introduction :

Le mode trajectoire permet à un automate ou un boîtier de commande externe de lancer des mouvements (jusqu'à 64, préenregistrés dans une table) à partir des entrées logiques du module d'extension ; Il est aussi possible de gérer directement ses trajectoires par Modbus ou CANopen.



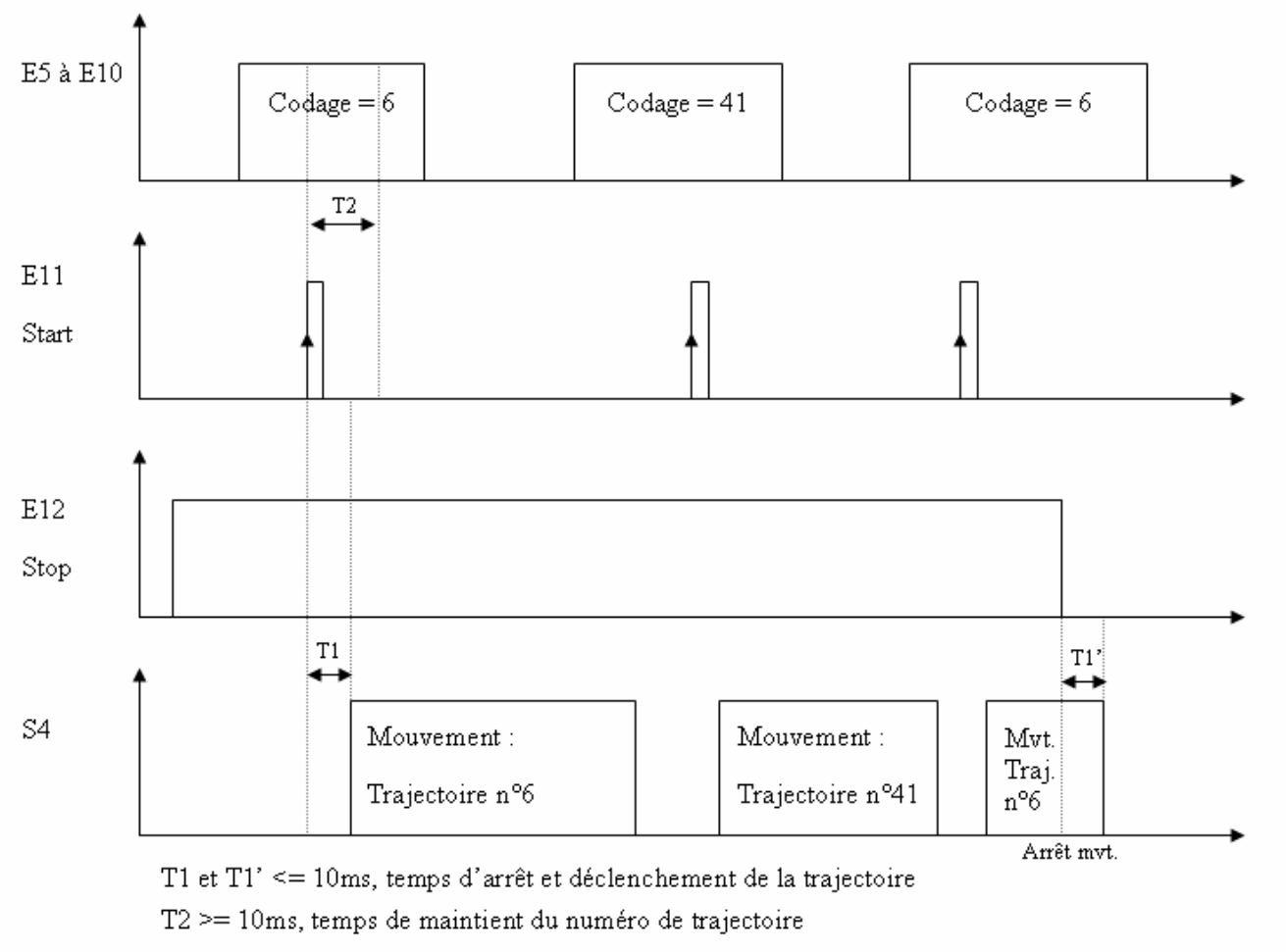
Pour chaque profil de trajectoire, on peut définir une vitesse, une accélération et une décélération. Tous ces paramètres sont stockés dans les 64 premières variables de type réel et entier long.

Si vous utilisez le iDPL en même temps que les trajectoires, la modification des variables : VR0 à VR63 ou VL0 à VL63 par les tâches iDPL modifiera aussi les trajectoires correspondantes.

5-2- Trajectoires par carte I/O

5-2-1- Fonctionnement avec carte I/O:

a) Chronogrammes :



b) Carte d'extension I/O :

- De E5 à E10 : 6 entrées pour le codage du numéro de trajectoire, avec E5 étant le bit de poids faible et E10 le bit de poids fort.
- E11 : entrée START sur front montant déclenchant le mouvement.
- E12 : entrée STOP, à niveau logique 1 en fonctionnement. Si passage à niveau logique 0, tout mouvement en cours s'arrête.
- S3 : sortie image de la prise d'origine : 0 si home non fait et 1 si fait
- S4 : sortie image du mouvement (MOVE_S) : 0 si axe à l'arrêt et 1 si axe en mouvement.

Attention : E_5 correspond à la 1ère entrée du module d'extension I/O

Si le filtrage des entrées a été activé, majoré les différents temps de la durée du filtrage.

c) Composition d'une trajectoire :

Chaque trajectoire est codée sur un réel et un entier long.

Ex : La trajectoire TRJ0 est codée sur VR0 et VL0

La trajectoire TRJ19 est codée sur VR19 et VL19

- La variable réelle contient la position de la trajectoire.
- L'entier long est divisé en 4 octets suivants :

1er octet : le mode (poids fort)

- 0 : absolu
- 1 : relatif
- 2 : infini +
- 3 : infini –
- 4 : home

2ième octet : la vitesse (en %)

3ième octet : l'accélération (en %)

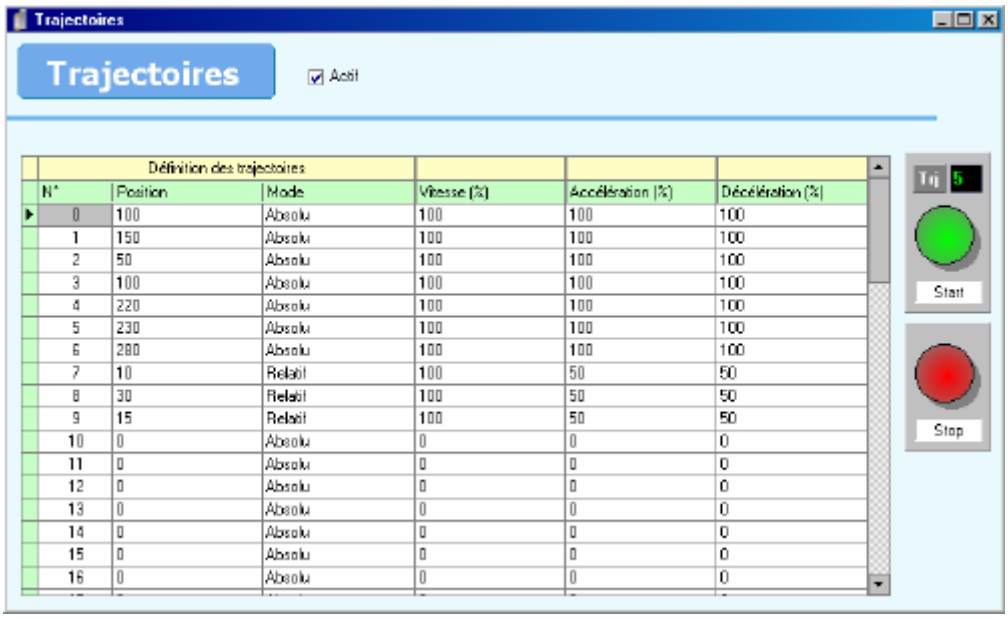
4ième octet : la décélération (poids faible) (en %)

5-2-2- Mise en oeuvre avec carte I/O:

a) Définition des trajectoires :

Pour avoir accès aux trajectoires, il faut que le variateur soit en mode position.

- Cliquer sur **Trajectoires** dans le menu **Motion Control** .
- Si le variateur est connecté au PC, ce dernier va chercher les trajectoires contenues dans le variateur et les affiches sinon il vous demande d'ouvrir un fichier de trajectoires ou d'en créer un.


Trajectoires
Trajectoires
Actif
Définition des trajectoires
N°
Position
Mode
Vitesse (%)
Accélération (%)
Décélération (%)
0 100 Absolu 100 100 100
1 150 Absolu 100 100 100
2 50 Absolu 100 100 100
3 100 Absolu 100 100 100
4 220 Absolu 100 100 100
5 230 Absolu 100 100 100
6 280 Absolu 100 100 100
7 10 Relatif 100 50 50
8 30 Relatif 100 50 50
9 15 Relatif 100 50 50
10 0 Absolu 0 0 0
11 0 Absolu 0 0 0
12 0 Absolu 0 0 0
13 0 Absolu 0 0 0
14 0 Absolu 0 0 0
15 0 Absolu 0 0 0
16 0 Absolu 0 0 0
Trj 5
Start
Stop

- Sélectionner trajectoires en mode standard.
- Pour chaque trajectoire vous devez entrer :
  1. une position
  2. un mode : absolue, relation, infini +, infini – ou home
  3. une vitesse en %
  4. une accélération en %
  5. une décélération en %

Toutes les valeurs saisies dépendent des **unités** et **profil de vitesse** entrés dans **Motion Control / Configuration.**

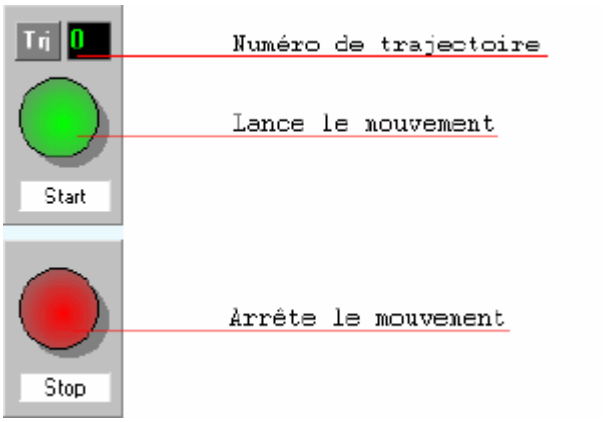
Pour exécuter une prise d'origine à partir des trajectoires :

1. Déclarer une trajectoire en mode HOME.
2. Paramétrer la prise d'origine dans **Motion Control / Home**.
3. Paramétrer l'entrée E4 en fonction **Home** dans **Paramètres \ E/S Logiques**, si vous utilisez un capteur de prise d'origine.

Sauver les trajectoires avec **Communication / Trajectoires / Sauver les trajectoires**.

b) Simulation des trajectoires :

Dans l'écran **Définition des trajectoires,** vous pouvez simuler les trajectoires saisies :



1. Vérifier que le variateur est asservi et que la case active est cochée.
2. Cliquer sur le numéro de la trajectoire à exécuter.
3. Appuyer sur START pour lancer la trajectoire.
4. Appuyer sur STOP si l'on souhaite arrêter le mouvement avant la fin.

c) Les fichiers TRJ :

- Il est possible d'enregistrer les trajectoires contenues dans le variateur vers un fichier .trj avec **Communication** / **Trajectoires** / **Recevoir les trajectoires.**
- De la même manière, il est possible de transférer les trajectoires contenues dans un fichier .trj vers le variateur avec **Communication** / **Trajectoires** / **Envoyer les trajectoires.**

5-3- Trajectoires par bus de communication

5-3-1- Fonctionnement par bus de communication:

Il est possible de simuler le mode trajectoire via le bus de communication en modifiant directement les paramètres trajectoires (voir le fichier ..\SERAD\iDpl1.12\Data\Modbus.htm).

a) Paramètres trajectoires :

- _PARAM_TRAJ_ACTIF permet de rendre actif le mode trajectoire (mettre à 2).
- _PARAM_TRAJ_SELECTION permet de sélectionner une trajectoire (de 0 à 63).
- _PARAM_TRAJ_START permet de démarrer la trajectoire sélectionnée.
- _PARAM_TRAJ_STOP permet d'arrêter la trajectoire en cours.

b) Composition d'une trajectoire :

Chaque trajectoire est codée sur un réel et un entier long.

Ex : La trajectoire TRJ0 est codée sur VR0 et VL0

La trajectoire TRJ19 est codée sur VR19 et VL19

- La variable réelle contient la position de la trajectoire.
- L'entier long est divisé en 4 octets suivants :

  1er octet : le mode (poids fort)

  - 0 : absolu
  - 1 : relatif
  - 2 : infini +
  - 3 : infini –
  - 4 : home

  2ième octet : la vitesse

  3ième octet : l'accélération

  4ième octet : la décélération (poids faible)

5-3-2- Mise en oeuvre par bus de communication:

Exemple de trajectoires par bus CAN:

Prog

```
'Démo Bitconnect CAN/ModBus/iDPL

'passage trajectoire(BitConnect) en mode Bus de Communication
WriteParam(2800h,01h)=2

WriteParam(6040h,00h)=0 'Disable drive
wait (readParam(6041h,00h)=0); 'attend que le drive soit disable
WriteParam(6040h,00h)=1 'Enable drive
wait (readParam(6041h,00h)=1); 'attend que le drive soit enable

'toute les trajectoires sont écrites dans
'la case 0 (VL0,VR0) avant chaque utilisation
WriteParam(2800h,04h)=0 'préselectionne la case 0

' === Prise d'origine ===
'---Ecriture de la trajectoire---
VR0=0'WriteParam(3400h,00h)=0 'position 0
VL100=4         'mode : Home
VL100=VL100 << 8
VL100=VL100+0   'vitesse : 0
VL100=VL100 << 8
VL100=VL100+0   'acceleration : 0
VL100=VL100 << 8
VL100=VL100+0   'deceleration : 0
VL0=VL100'WriteParam(3300h,00h)=VL100 ' options
```

```
'---Lancement---
WriteParam(2800h,02h)=1
'attend que la prise d'origine soit effectuée
repeat
    VI100=ReadParam(6510h,06h)
    VI100=VI100 and 2
until VI100<>0

' === Trajectoire 1 ===
'---Ecriture de la trajectoire---
VR0=-5'WriteParam(3400h,00h)=-500 'position -5 (tient compte de la précision du DPL)
VL100=0         'mode : Absolu
VL100=VL100 << 8
VL100=VL100+20   'vitesse : 20
VL100=VL100 << 8
VL100=VL100+100   'acceleration : 100
VL100=VL100 << 8
VL100=VL100+100   'deceleration : 100
VL0=VL100'WriteParam(3300h,00h)=VL100 ' options
'---Lancement---
WriteParam(2800h,02h)=1
'attend que le mouvement soit terminé
repeat
    VI100=ReadParam(6510h,06h)
    VI100=VI100 and 1
until VI100=0

'---Ecriture de la trajectoire---
VR0=-1'WriteParam(3400h,00h)=-100 'position -1 (tient compte de la précision du DPL)
```

```
VL100=1         'mode : Relatif
VL100=VL100 << 8
VL100=VL100+10   'vitesse : 10
VL100=VL100 << 8
VL100=VL100+100   'acceleration : 100
VL100=VL100 << 8
VL100=VL100+100   'deceleration : 100
VL0=VL100'WriteParam(3300h,00h)=VL100 ' options
'---Lancement---
WriteParam(2800h,02h)=1
'attend que le mouvement soit terminé
repeat
    VI100=ReadParam(6510h,06h)
    VI100=VI100 and 1
until VI100=0

'---Ecriture de la trajectoire---
VR0=2.5'WriteParam(3400h,00h)=250 'position 2.5 (tient compte de la précision du DPL)
VL100=0         'mode : Absolu
VL100=VL100 << 8
VL100=VL100+30   'vitesse : 30
VL100=VL100 << 8
VL100=VL100+100   'acceleration : 100
VL100=VL100 << 8
VL100=VL100+100   'deceleration : 100
VL0=VL100'WriteParam(3300h,00h)=VL100 'options
'---Lancement---
WriteParam(2800h,02h)=1
'attend d'être passé en 0
```

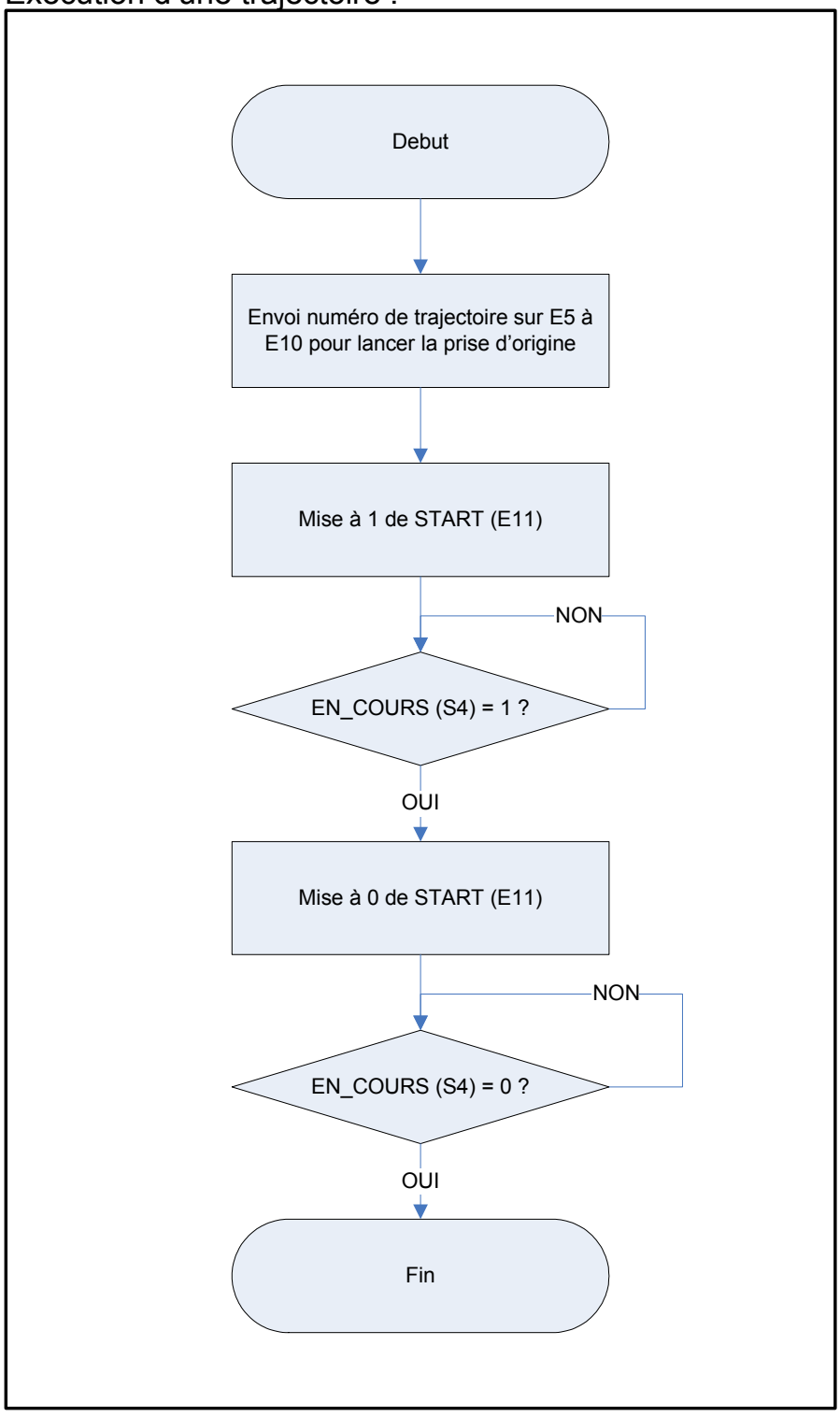
```
    repeat
        VL100=ReadParam(6064h,00h) 'position actuelle
        VR100=VL100
        VR100=VR100/100 'divise par la précision du DPL
    until VR100>0
    WriteParam(2800h,03h)=1 'stoppe le muvement
    'attend que le mouvement soit terminé
    repeat
        VI100=ReadParam(6510h,06h)
        VI100=VI100 and 1
    until VI100=0

    '---Ecriture de la trajectoire---
    VR0=0'WriteParam(3400h,00h)=0 'position 0
    VL100=2         'mode : Infini +
    VL100=VL100 << 8
    VL100=VL100+30   'vitesse : 30
    VL100=VL100 << 8
    VL100=VL100+100   'acceleration : 100
    VL100=VL100 << 8
    VL100=VL100+100   'deceleration : 100
    VL0=VL100'WriteParam(3300h,00h)=VL100 'options
    '---Lancement---
    WriteParam(2800h,02h)=1
    delay (1000)
    WriteParam(2800h,03h)=1 'stoppe le mouvement

    halt 1
EndProg
```

5-4- Trajectoires avec carte I/O en mode avancé :

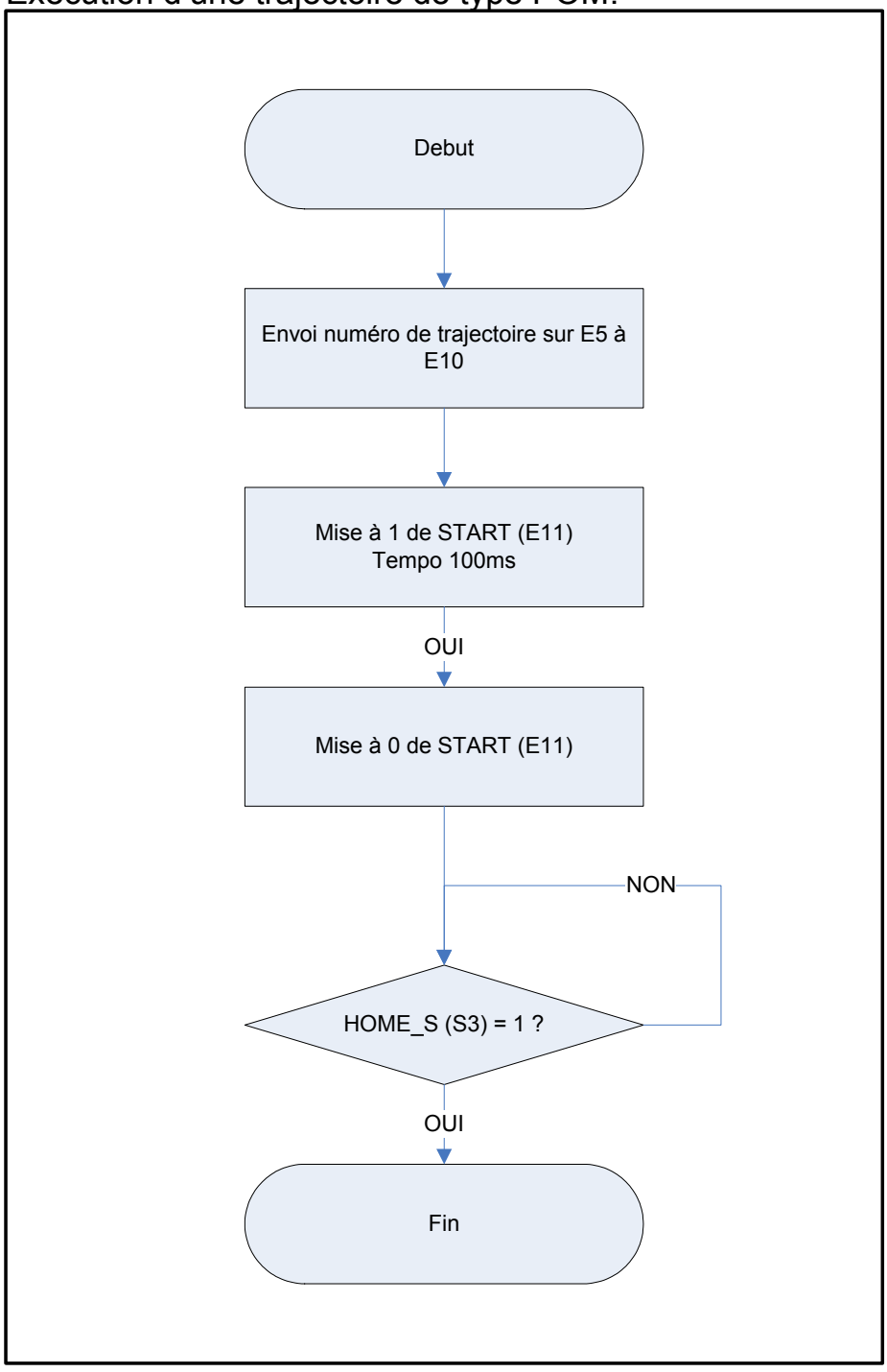
5-4-1- Fonctionnement par carte I/O en mode avancé :

a) Organigrammes :

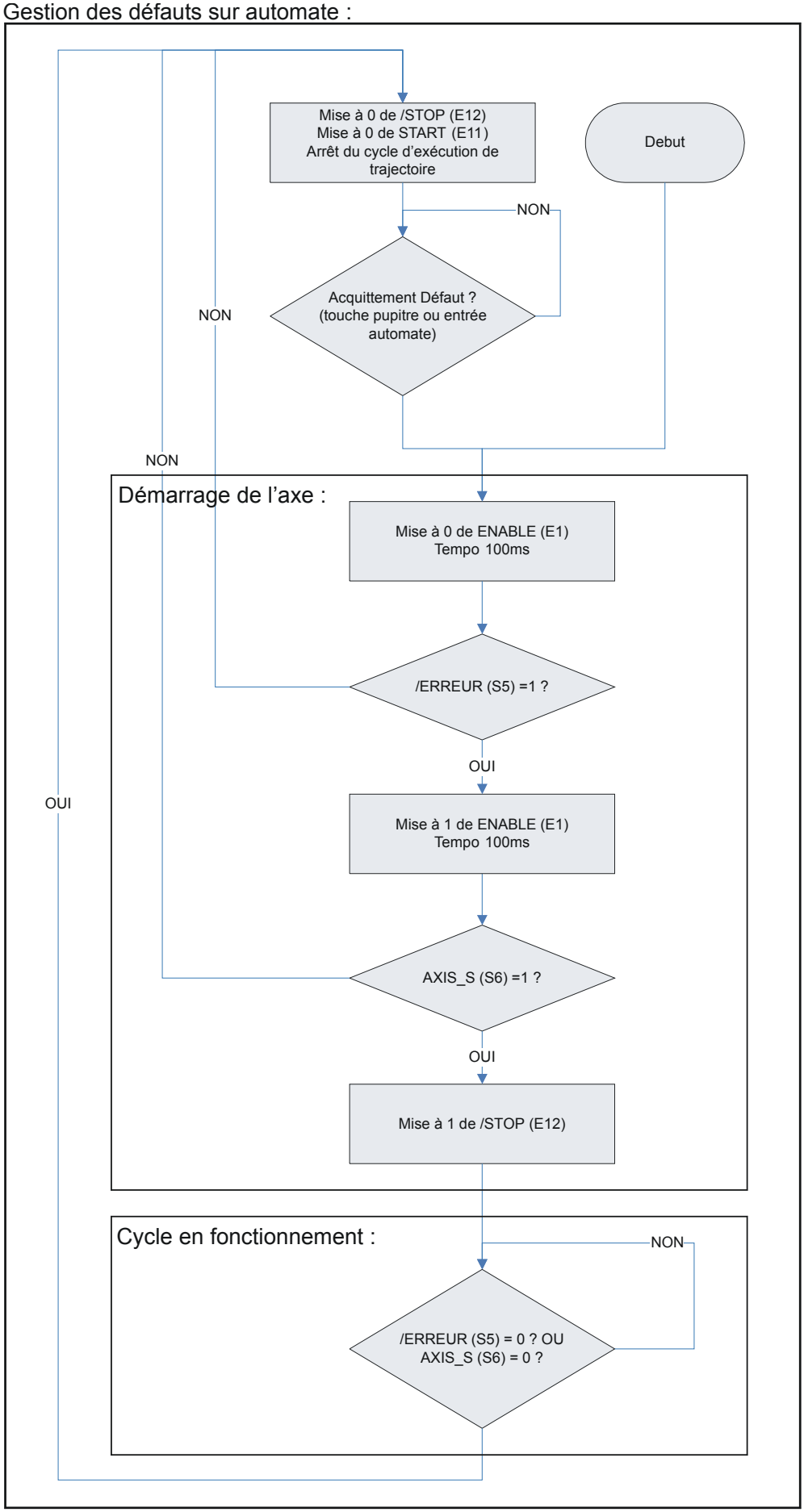
Exécution d'une trajectoire :



Exécution d'une trajectoire de type POM:



Gestion des défauts sur automate :



b) De base :

E1 : entrée ENABLE permet d'asservir le variateur sur front montant et de désasservir l'axe sur état 0 (l'entée 1 doit être déclarée en fonction VALIDATION dans Paramètres \ E/S Logiques).

c) Sur extension I/O :

De E5 à E10 : 6 entrées pour le codage du numéro de trajectoire, avec E5 étant le bit de poids faible et E10 le bit de poids fort.

E11 : entrée START sur front montant déclenchant le mouvement.

E12 : entrée STOP, à niveau logique 1 en fonctionnement. Si passage à niveau logique 0, tout mouvement en cours s'arrête.

S3 : sortie image de la prise d'origine (HOME_S) : 0 si home non fait et 1 si fait

S4 : sortie image du mouvement (EN_COURS) : 0 si axe à l'arrêt et 1 si axe en mouvement.

S5 : sortie image erreur de trajectoire (/ERREUR) : 0 si erreur de trajectoire et 1 si pas de défaut.

S6 : sortie image de l'état de l'asservissement (AXIS_S).

Attention : E_5 correspond à la 1ère entrée du module d'extension I/O

Si le filtrage des entrées a été activé, majoré les différents temps de la durée du filtrage.

d) Composition d'une trajectoire :

Chaque trajectoire est codée sur un réel et un entier long.

Ex : La trajectoire TRJ0 est codée sur VR0 et VL0

La trajectoire TRJ19 est codée sur VR19 et VL19

- La variable réelle contient la position de la trajectoire.
- L'entier long est divisé en 4 octets suivants :

1er octet : le mode (poids fort)

- 1 : relatif
- 2 : infini +
- 3 : infini –
- 4 : home

2ième octet : la vitesse (en %)

3ième octet : l'accélération (en %)

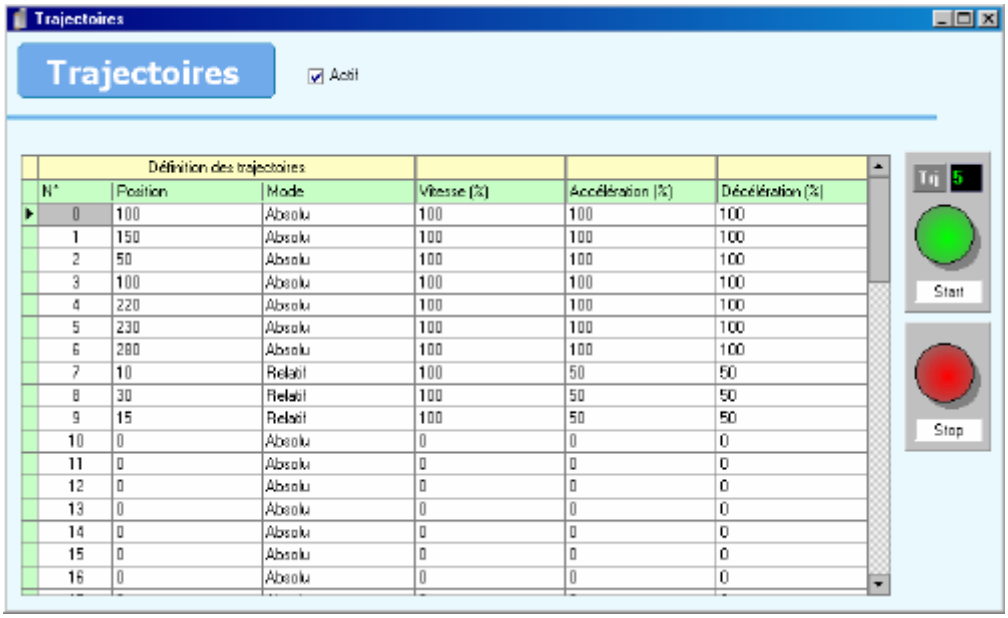
4ième octet : la décélération (poids faible) (en %)

5-4-2- Mise en oeuvre avec carte I/O en mode avancé :

a) Définition des trajectoires :

Pour avoir accès aux trajectoires, il faut que le variateur soit en mode position.

- Cliquer sur **Trajectoires** dans le menu **Motion Control**.
- Si le variateur est connecté au PC, ce dernier va chercher les trajectoires contenues dans le variateur et les affiches sinon il vous demande d'ouvrir un fichier de trajectoires ou d'en créer un.



- Sélectionner le mode trajectoire en mode avancé.
- Pour chaque trajectoire vous devez entrer :

1. une position
2. un mode : absolue, relation, infini +, infini – ou home
3. une vitesse en %
4. une accélération en %
5. une décélération en %

Toutes les valeurs saisies dépendent des **unités** et **profil de vitesse** entrés dans **Motion Control / Configuration.**

Une prise d'origine exécutée par une trajectoire utilisera les paramètres entrés dans **Motion Control / Configuration / Home**

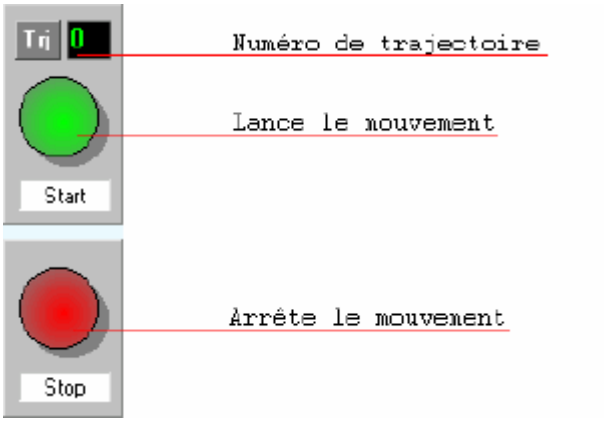
Pour exécuter une prise d'origine à partir des trajectoires :

1. Déclarer une trajectoire en mode HOME.
2. Paramétrer la prise d'origine dans **Motion Control / Home**.
3. Paramétrer l'entrée E4 en fonction **Home** dans **Paramètres \ E/S Logiques**, si vous utilisez un capteur de prise d'origine.

Sauver les trajectoires avec **Communication** / **Trajectoires** / **Sauver les trajectoires**.

b) Simulation des trajectoires :

Dans l'écran **Définition des trajectoires,** vous pouvez simuler les trajectoires saisies :



1. Vérifier que le variateur est asservi et que la case active est cochée.
2. Cliquer sur le numéro de la trajectoire à exécuter.
3. Appuyer sur START pour lancer la trajectoire.
4. Appuyer sur STOP si l'on souhaite arrêter le mouvement avant la fin.

c) Les fichiers TRJ :

- Il est possible d'enregistrer les trajectoires contenues dans le variateur vers un fichier .trj avec **Communication** / **Trajectoires** / **Recevoir les trajectoires.**
- De la même manière, il est possible de transférer les trajectoires contenues dans un fichier .trj vers le variateur avec **Communication** / **Trajectoires** / **Envoyer les trajectoires.**

6- Langage de programmation iDPL

6-1- Introduction

6-1-1- Introduction

- Le langage iDPL (intelligent Drive Programming Language) est un outil de programmation puissant et simple à utiliser. Il offre une architecture structurée rencontrée sur les langages de haut niveau. Pour une programmation flexible, ce langage est géré par un noyau temps réel multitâches, utilisant des instructions pseudo-basics et contenant également toutes les fonctions de contrôle de mouvement et d'automate.
- Le langage intègre aussi la gestion de données sous la forme de variables.
- Un projet développé à partir du iDPL peut contenir jusqu'à 4 tâches fonctionnant en parallèle. Chaque tâche possède un niveau de priorité et est écrite en basic.
- Le variateur possède aussi une zone FRAM de 4096 mots pour sauvegarder des données sous la forme de variables ou de cames.

6-1-2- Affectation du plan mémoire

Affectation de la mémoire FLASH

Zone réservée au système :

- Boot
- Système d'exploitation (Firmware)

Paramètres système

Valeurs variables initialisées :

- VR0 à VR63
- VL0 à VL63

Programme BASIC :

- 14 Ko

4 tâches motion-basic

Affectation de la mémoire RAM

Zone réservée au système :

- Système d'exploitation

256 variables de type réel :

- 1Ko
- de VR0 à VR255

256 variables de type entier long signé :

- 1 Ko
- VL0 à VL255

256 variables de type entier non signé :

- 512 octets
- VI0 à VI255

256 variables de type octet :

- 256 octets
- VB0 à VB255

256 variables de type flag :

- 32 octets
- VF0 à VF255

Affectation de la mémoire FRAM

Zone sauvegarde de données :

- Variable entier
- Variable entier long
- Variable réel
- Table de came

6-2- Les variables

6-2-1- Variables

Toute variable est globale et peut être utilisée par plusieurs tâches.

Elle peut aussi être traitée comme un tableau (notion d'indexage).

On peut attribuer un nom à une variable à partir de l'écran **Langage iDPL** / **Déclarations / Variables** et l'utiliser dans les tâches iDPL.

Ex : Position = POS_S

Les variables sont numérotées de 0 à 255.

Tableau récapitulatif des différents types :

Type	Valeur	Occupation mémoire	Exemple
Flag (Bit)	1/0, On/Off ou True/False	1 bit à l'intérieur d'un mot	VF0 à VF255
Octet (Byte)	0 à 255	1 octet	VB0 à VB255
Entier (Integer)	0 à 65535	2 octets non signés	VI0 à VI255
Entier long (Long)	+/- 2 147 483 647	4 octets signés	VL0 à VL255
Réel (Real)	+/- 2 147 483 647	4 octets signés	VR0 à VR255

Tous les calcules doivent être du type **<Variable1> = <Variable2> <Expression> <Variable3 ou Constante>**

Avec <Variable1> de même type que <Variable2> et <Variable3> de type inférieur ou égale à <Variable1>

Ex : VR0 = VR1 * 100

VR0 = VR1 * VR2

VL0 = VL0 * VB0

Il est possible d'utiliser des variables indexées pour se déplacer dans un tableau.

VL22 = VL0[7] 'est équivalent VL22 = VL7

VL23 = VL2[9] 'est équivalent VL23 = VL11

VB3 = 9

VL24 = VL5[VB3] ‘est équivalent VL24 = VL14

Atttention : On peut utiliser les tableaux seulement pour les affectations

Exemple 1 : VR0=VR0 [VB1]

STTA = VR0

Exemple 2 : VR0=VR0 [VB2]

VL0=VL0 [VB3]

VR0= VR0 * VL0

Les variables du type réel sont des entiers longs signés que l’on divise par un coefficient du type 1, 0.1, 0.01 … (type réel à virgule fixe)

Pour changer ce coefficient, aller dans **Option** -> **Langage iDPL** -> **Compilateur**, le projet doit être recompiler pour tenir compte des changements.

6-2-2- Conversions de type de variables

Pour convertir un type de données en un autre, il suffit de faire une affectation :

- Type flag :

```
VB1 = VF0
VI1 = VF0
VL1 = VF0
VR1 = VF0
```

- Type octet

```
VF2 = VB0        ‘ VF2 est égale au 1er bit de poids faible de VB0
VI2 = VB0
VL2 = VB0
VR2 = VB0
```

- Type entier

```
VF3 = VI0        ‘ VF3 est égale au 1er bit de poids faible de VI0
VB3 = VI0        ‘ VB3 est égale aux 8 premiers bits de poids faible de VI0
VL3 = VI0
VR3 = VI0
```

- Type entier long

```
VF4 = VL0       ‘ VF4 est égale au 1er bit de poids faible de VL0
VB4 = VL0       ‘ VB4 est égale aux 8 premiers bits de poids faible de VI0
VI4 = VL0       ‘ VI4 est égale aux 16 premiers bits de poids faible de VL0
VR4 = VL0
```

- Type réel

```
VF5 = VR0       ‘ VF5 est égale au 1er bit de poids faible de la partie entière de VR0
VB5 = VR0       ‘ VB5 est égale aux 8 premiers bits de poids faible de la partie entière de VR0
VI5 = VR0       ‘ VI5 est égale aux 16 premiers bits de poids faible de la partie entière de VR0
VL5 = VR0       ‘ VL5 est égale à la partie entière de VR0
```

6-2-3- Notation numériques

Les valeurs numériques peuvent être exprimées en décimal, en hexadécimal, en binaire.

Exemple :

```
VB0=254             ‘ notation décimale
VB1=0FEh            ‘ notation hexadécimale
VB2=11111110b       ‘ notation binaire
```

6-2-4- Variables globales sauvegardées

Certaines variables globales (VR0 à VR63, VL0 à VL63) peuvent être sauvegardées afin d’être initialisées après une coupure d’alimentation (24V) ou redémarrage du variateur.

a) SAVEVARIABLE – Permet de sauvegarder les variables

Syntaxe : SAVEVARIABLE

Description : Les variables en RAM VR0 à VR63, VL0 à VL63 sont sauvegardées en mémoire FLASH.

Le variateur passe automatiquement en AXIS OFF

Remarque : La FLASH à une durée de vie de 5000 cycles d'écriture.

Attention : Consulter notre service technique avant l'utilisation de cette instruction sous peine de dégradation prématurée de la mémoire FLASH

L'utilisation des instructions SAVEPARAM et SAVEVARIABLE fausse la base de temps et provoque l'arrêt de l'envoie de la position CAN.

b) LOADVARIABLE - Permet de transférer les variables sauvegardés

Syntaxe : LOADVARIABLE

Description : Permet de transférer dans la mémoire de travail, les variables VR0 à VR63 et VL0 à VL63 sauvegardés de la mémoire FLASH.

6-3- Les données sauvegardées

6-3-1- Les données sauvegardées

a) 4096 mots (16 bits) en FRAM :

Avantage de la mémoire FRAM :

- Nombre de cycles de lecture et écriture illimité
- Sauvegarde des données après coupure d'alimentation

Grâce à ses caractéristiques, on peut utiliser la mémoire FRAM comme zone de variables sauvegardées.Elle permet de stocker des variables de type entier, entier long, réel et tableaux de cames.

N° de mot	Adresse	Fonction	
Mot n°1	Adresse 0	Variable 1 (entier)	
Mot n°1	Adresse 1	Variable 2 (entier)	
Mot n°1	Adresse 2	Variable 3 (réel)	
	Adresse 3		
Mot n°9	Adresse 8		
Mot n°10	Adresse 9	Came 1	Position maître

Mot n°11	Adresse 10	– point 1	
Mot n°12	Adresse 11		Position esclave
Mot n°13	Adresse 12		
Mot n°14	Adresse 13		Tangente maître
Mot n°15	Adresse 14		
Mot n°16	Adresse 15		Tangente esclave
Mot n°17	Adresse 16		
Mot n°18	Adresse 17	Came 1 – point 2	Position maître
Mot n°19	Adresse 18		
Mot n°4095	Adresse 4094	Vide	
Mot n°4096	Adresse 4095	Vide	

b) Lecture/écriture d'un entier :

Ecriture : WRITEI (<Adresse>) = <VIn ou valeur>

Lecture : <VIn> = READI (<Adresse>)

Limites : <Adresse> : de 0 à 4095

n de 0 à 255

c) Lecture/écriture d'un entier long :

Ecriture : WRITEL (<Adresse>) = <VLn ou valeur>

Lecture : <VLn> = READL (<Adresse>)

Limites : <Adresse> : de 0 à 4094

n de 0 à 255

Attention : La lecture et l'écriture d'un entier long utilisent 2 adresses mémoires consécutives (adresse et adresse+1).

d) Lecture/écriture d'un réel :

Ecriture : WRITER (<Adresse>) = <VRn ou valeur>

Lecture : <VRn> = READR (<Adresse>)

Limites : <Adresse> : de 0 à 4094

n de 0 à 255

Attention : La lecture et l'écriture d'un réel utilisent 2 adresses mémoires consécutives (adresse et adresse+1).

e) Lecture/écriture d'un tableau de came :

Voir chapitre sur les cames dans la partie contrôle de mouvement

Vérifier que vos tables de cames et vos données sauvegardées n'utilisent pas les mêmes adresses afin de ne pas avoir de phénomènes aléatoires (pertes de données) lors de l'exécution de vos cames ou de vos tâches.

6-4- Les paramètres

6-4-1- Les paramètres

Il est possible à partir d'une tâche iDPL de modifier les paramètres du variateur afin de changer mode de fonctionnement (mode couple, vitesse ou position), le rôle d'une entrée, un gain de régulation ...

La liste des paramètres se trouve dans le fichier ...\iDPL\DATA\Modbus.htm

a) READPARAM - Lecture d'un paramètre

Syntaxe : <Variable> = READPARAM (<Index>, <Sub-Index>)

Types acceptés : <Variable> du type entier long

<Index> de type entier

<Sub-Index> de type octet

Description : Cette fonction permet de lire via le bus CANopen, les paramètres du variateur.

Exemple : VL0 = READPARAM(8448,1) 'Renvoie le numéro du défaut du variateur

b) WRITEPARAM – Ecriture d'un paramètre

Syntaxe : READPARAM (<Index>, <Sub-Index>) = <Variable>

Types acceptés : <Variable> du type entier long

<Index> de type entier

<Sub-Index> de type octet

Description : Cette fonction permet de lire via le bus CANopen, les paramètres du variateur.

Exemple : WRITEPARAM(9984,6) = 1 'Active le modulo sur l'axe

c) SAVEPARAM - Permet de sauvegarder les paramètres du variateur

Syntaxe : SAVEPARAM

Description : Les paramètres du variateurs en RAM EXTERNE sont sauvegardés en mémoire XFLASH.

Remarque : La FLASH à une durée de vie de 5000 cycles d'écriture.

Attention : Consulter notre service technique avant l'utilisation de cette instruction sous peine de dégradation prématurée de la mémoire FLASH

d) LOADPARAM – Permet de recharger les paramètres du variateur

Syntaxe : LOADPARAM

Description : Permet de transférer dans la mémoire de travail RAM, les paramètres sauvegardés de la mémoire FLASH.

6-5- Les Tâches

6-5-1- Principes du multitâches

Le moniteur temps réel multitâches gère jusqu'à 4 tâches en parallèle :

Le multitâche bascule de la tâche courante vers la tâche suivante si :

↳ Le temps passé dans la tâche dépasse le *temps de vieillissement*. Ce temps est paramétrable à partir du menu Options / Langage iDPL / Compilateur. Il est nécessaire de recompiler les tâches après une modification.

↳ Rencontre d'une instruction bloquante :

- Wait, Delay
- Mova, Movr, Stop, Home

↳ rencontre de l'instruction NEXTTASK

En règle générale, une tâche courte permettra de traiter des événements plus rapides qu'une tâche longue.

6-5-2- Priorité des tâches

Dans un projet iDPL, on intègre un niveau de priorité pour les tâches :

On peut avoir une tâche de priorité haute et les autres de priorité normale.

La tâche de priorité haute occupe un temps de vieillissements sur deux :

Nbr. tâches	Répartition du temps d'exécution	Exemple avec la tâche 1 en priorité haute
	Tâche haute | Tâches normales	Cycle d'exécution des tâches
1	Pas de changement	1
2	75% - 25%	1-1-1-2
3	66% - 33%	1-1-1-2-1-3
4	62,5% - 37,5%	1-1-1-2-1-3-1-4

6-5-3- Gestion des tâches

Chaque tâche possède un mode de démarrage qui a été paramétré lors de sa création :

↳ Démarrage automatique : à chaque démarrage du variateur, la tâche est lancée automatiquement.

↳ Démarrage manuel : la tâche n'est pas lancée automatiquement.

Un projet doit au moins contenir une tâche avec démarrage automatique. Il est conseillé d'avoir une seule tâche dans laquelle on écrit toute la partie initialisation de l'application et ensuite on lance les autres tâches.

On dispose de 5 instructions pour gérer les tâches :

↳ Run : lancement d'une tâche qui est à l'arrêt.

↳ Suspend : suspension (pause) d'une tâche en cours d'exécution.

↳ Continue : reprise de l'exécution d'une tâche suspendue là où elle c'était arrêtée.

↳ Halt : arrêt d'une tâche en cours d'exécution.

↳ Status : indique l'état de la tâche.

Exemple :

Tâche 1	Tâche 2
Prog	Prog
.....	.....
Run 2	If VR1 = 0 Halt 2
Wait Status(2)=0	.....
....	End Prog
End Prog	

Attention : L'arrêt ou la suspension de la tâche n'affecte pas les mouvements lancés par celle-ci

Exemple :

Tâche 1	Tâche 2
Prog	Prog
.....	.....
If VF=0 Goto CYCLE_PROD	Mova(1000)
Halt 2	Out(6)=1
Stop	Mova(2000)
CYCLE_PROD	.....

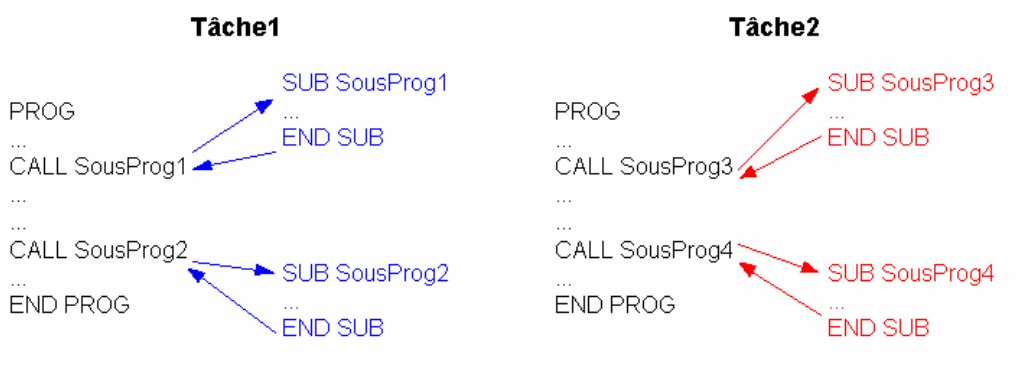
....

End Prog

End Prog

6-5-4- Structure d'une tâche basic

Chaque tâche est constituée d'un programme principal défini par les mots clé PROG et END PROG et par des sous programmes sous forme de structure SUB .. END SUB. Par exemple :



a) Programme principal

Le programme principal d'une tâche peut appeler tous les sous programmes de la tâche mais ne peut pas appeler les sous programmes d'une autre tâche. Une tâche correspond à un fichier. Dans l'exemple précédent, la tâche1 peut appeler les sous-programmes SousProg1 et SousProg2 mais ne peut pas appeler les sous- programmes SousProg3 et SousProg4. Un sous programme d'une tâche peut également appeler un autre sous-programme de la même tâche.

Une seule structure PROG ... END PROG doit être utilisée par tâche. Elle peut apparaître à n'importe quel endroit.

Pendant l'exécution de la tâche, la rencontre du mot clé END PROG provoque un branchement de celle-ci en PROG.

b) Sous-programmes

Un sous-programme doit être déclaré par une procédure SUB...END SUB. Il peut être placé avant ou après le programme principal.

Pour appeler un sous-programme, vous devez utiliser la fonction CALL. Le sous-programme appelé doit être dans la même tâche.

Après l'appel du sous-programme, son exécution et son retour, la tâche continue automatiquement à l'instruction qui suit l'appel du sous-programme. Le système sort d'un sous programme lorsqu'il rencontre l'instruction END SUB ou EXIT SUB. Par exemple :

```
SUB Calcul
VR2=0
IF VR1=0 EXIT SUB          ' Si VR1 est égal à zéro la division est impossible
VR2=VR10/VR1    ' Division
END SUB
```

Un sous-programme peut être appelé partout dans le programme mais ne peut s'appeler lui-même. Si des données sont utilisées dans le programme et dans des sous programmes, il est recommandé d'utiliser des variables bien spécifiques. En fait, toutes les variables peuvent être modifiées par un sous-programme, vous pouvez donc utiliser ces variables spécifiques dans chaque sous-programme en les affectant simplement avant l'appel. Par exemple :

```
...
VR100=VR1
VR101=VR18
CALL Divise
IF VR102>10  Goto ...
...


SUB Divise
VR102=0
IF VR100=0 EXIT SUB
VR102=VR100/VR101
END SUB
```

c) Branchement à une étiquette

L'instruction GOTO sert à effectuer un saut à une adresse représentée par une étiquette. Une étiquette est composée d'un nom terminé par ":". Si l'instruction GOTO se trouve à l'intérieur d'une structure de sous-programme SUB...END SUB, l'étiquette doit se trouver dans cette même structure.

Un branchement avec l'instruction GOTO peut être effectué indifféremment vers l'avant ou l'arrière du programme. Par exemple:

```
GOTO Label1
...
Label1:
...
```

d) Opérateurs

Les expressions sont composées d'opérateurs et d'opérandes. En Basic presque tous les opérateurs sont binaires, c'est à dire qu'ils utilisent deux opérandes. Les opérateurs n'utilisant qu'un opérande sont qualifiés d'unaires. Les opérateurs binaires utilisent les formes algébriques communes, par exemple A + B. Les opérateurs unaires s'écrivent toujours avant leurs opérandes, par exemple : NOT A. Dans des expressions complexes les règles de priorité suivantes enlèvent toute ambiguïté sur l'ordre des opérateurs.

Opérateur	Priorité	Type
NOT	Première (Haute)	Unaire
*, /, DIV, MOD, ,AND , <<, >>	Seconde	Multiplication
+, -, OR, XOR	Troisième	Addition
=, <>, <, >, <=, >=	Quatrième (Basse)	Comparaison

Dans une ligne programme, un seul opérateur pourra être traité.

(a) Opérateurs arithmétiques

L'opérateur 'NOT' est un opérateur unaire. Les opérateurs + et - sont employés comme des opérateurs unaires ou des opérateurs binaires. Les autres sont uniquement binaires.

Un opérateur unaire ne possède qu'un paramètre.

Par exemple : NOT <Expression>

Un opérateur binaire demande deux paramètres.

Par exemple : <Expression1> * <Expression2>

(b) Opérateurs binaires :

Opérateur	Opération	Type de l'opérande	Type
+	Addition	Octet, Entier, Entier long ou réel	Type de l'opérande
-	Soustraction	Octet, Entier, Entier long ou réel	Type de l'opérande
*	Multiplication	Octet, Entier, Entier long ou réel	Type de l'opérande
/	Division	Octet, Entier, Entier long ou réel	Réel
DIV	Division d'entiers	Octet, Entier, Entier long	Type de l'opérande
MOD	Modulo	Octet, Entier, Entier long	Type de l'opérande

(c) Opérateurs unaires :

Opérateur	Opération	Type de l'opérande	Type
+	Même signe	Octet, Entier, Entier long ou réel	Type de l'opérande
-	Inversion de signe	Octet, Entier, Entier long ou réel	Type de l'opérande

(d) Opérateurs logiques :

Opérateur	Opération	Type de l'opérande	Type
NOT	Négation binaire	Octet, Entier	Type de l'opérande
AND	ET logique	Octet, Entier	Type de l'opérande
OR	OU logique	Octet, Entier	Type de l'opérande
XOR	OU exclusif	Octet, Entier	Type de l'opérande
>>	Décalage à droite	Octet, Entier	Type de l'opérande
<<	Décalage à gauche	Octet, Entier	Type de l'opérande

(e) Opérateurs sur bits :

Opérateur	Opération	Type de l'opérande	Type
+	Même signe	Octet, Entier, Entier long ou réel	Type de l'opérande
-	Inversion de signe	Octet, Entier, Entier long ou réel	Type de l'opérande

(f) Opérateurs de relation :

Opérateur	Opération	Type de l'opérande	Type
=	Egal	Octet, Entier, Entier long, Réel ou chaîne de caractères	Bit
<>	Différent de	Octet, Entier, Entier long, Réel ou chaîne de caractères	Bit
<	Inférieur à	Octet, Entier, Entier long, Réel ou chaîne de caractères	Bit
>	Supérieur à	Octet, Entier, Entier long, Réel ou chaîne de caractères	Bit
<=	Inférieur ou égal à	Octet, Entier, Entier long, Réel ou chaîne de caractères	Bit
>=	Supérieur ou égal à	Octet, Entier, Entier long, Réel ou chaîne de caractères	Bit

e) Tests

Les instructions conditionnelles sont un moyen pratique d'exécuter ou non un groupe d'instructions selon qu'une condition est vraie ou fausse :

IF <Expression> GOTO <Etiquette>

…

Etiquette:

…

Ou

IF <Expression> THEN

```
    <Instruction1>
    ...
END IF
```

Ou

```
IF <Expression> THEN
    <Instruction1>
    ...
ELSE
    <Instruction2>
    ...
END IF
```

<Expression> doit être une valeur de type bit. Si <Expression> est vraie alors un saut à <Etiquette> est exécutée. Si <Expression> est fausse, le programme passe directement à la ligne suivante.

Exemple :

```
VEL%=100                          ' Vitesse rapide
STTA=2000                         ' Départ de l'axe à la position absolue 2000
MOVE_ON:
IF POS_S <1000 GOTO SUITE_VEL     'Si la position est supérieure ou égale à 1000 alors
VEL%=50                           ' la vitesse est diminuée de moitié.
SUITE_VEL:
IF POS_S<1500 GOTO SUITE_OUT      'Si la position est supérieure ou égale à 1500 alors
OUT(9)=1                          'la sortie 9 est activée.
SUITE_OUT:
IF MOVE_S=1 GOTO MOVE_ON          'Reboucle tant que le mouvement n'est pas fini.
…
```

f) Boucles

L'instruction REPEAT permet l'exécution répétée d'une ou plusieurs instructions selon la valeur d'une expression.

La syntaxe de l'instruction REPEAT est la suivante :

REPEAT

<Instructions>

UNTIL <Expression>

<Expression> doit être une valeur de type bit, si <Expression> est VRAIE avant la structure REPEAT, la boucle est effectuée une fois. <Instructions> sont exécutées jusqu'à ce que <Expression> soit vraie.

Par exemple :

```
VEL% = 100                      ' Vitesse rapide
STTA = 2000                     ' Start absolu en position 2000
REPEAT
        VR0 = POS_S
        IF VR0>1000 THEN
                VEL% =50        ' Vitesse lente à la moitié
        END IF                  ' de la distance
        UNTIL NOT MOVE_S        ' reboucler jusqu'à ce que
                                ' le moteur soit arrêté
```

7- Programmation du contrôle de mouvement

7-1- Introduction

Le variateur peut gérer un axe servo, une entrée maître et une sortie émulation.

Le logiciel iDPL contient de nombreuses instructions évoluées pour le contrôle de mouvement : positionnement, arbre électrique, superposition de mouvement, mouvements synchronisés ...

Les limites du compteur de position sont de ± 2 147 483 647 tour moteur

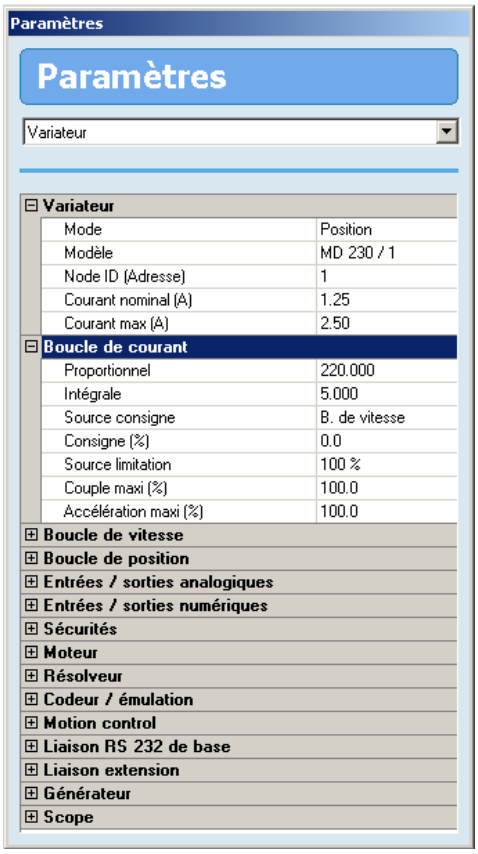
Il est possible d'inverser le sens moteur en boucle de position à partir de la liste de paramètres : Motion control / Inversion sens moteur.

7-2- Paramétrage d'un axe

7-2-1- Réglage d'un axe

Un axe doit être paramétré avant de pouvoir l'utiliser.

L'accès aux paramètres se fait à partir du menu **Paramètre** ou par accès direct grâce à la fenêtre « paramètres ».



A) Régulation

Il est conseillé d'utiliser la bibliothèque de paramètre moteur afin de calibrer les boucles de régulation nécessaire au bon fonctionnement du moteur, pour plus d'information voir le chapitre 4.

B) Erreur de poursuite maxi

Dès qu'un axe passe en mode asservi, il est contrôlé à tout moment : à l'arrêt, en mouvement.

Si la différence entre sa position théorique calculée et sa position réelle donnée par le retour résolveur est supérieure à l'erreur de poursuite maxi, le système passe l'axe servo en mode non asservi et ouvre le contact de le sortie « variateur prêt » (sauf si utilisation de l'instruction SECURITY).

Le réglage de cette valeur est très importante : une valeur trop petite entraîne des arrêts intempestifs sur l'axe, une valeur trop grande influe sur la sécurité des organes électriques et mécaniques.

Rentrer dans le champ «Erreur de poursuite maxi» de la fenêtre **Paramètre \ sécurités \ position**, la valeur adéquate (cette valeur est dans l'unité sélectionnée).

C) Fenêtre de position

Lorsque l'on envoie un axe à une position, la variateur considère que le mouvement est terminé quand le profil théorique de la trajectoire est exécuté et que la position réelle est comprise entre +/- la fenêtre de position. Par exemple, sur une machine de perçage où l'on recherche une précision de +/- 0.1mm, on réglera la fenêtre à cette valeur.

Rentrez dans le champ «Fenêtre de position » de la fenêtre **Paramètre \ sécurités \ position**, la valeur de précision recherchée (cette valeur est dans l'unité sélectionnée).

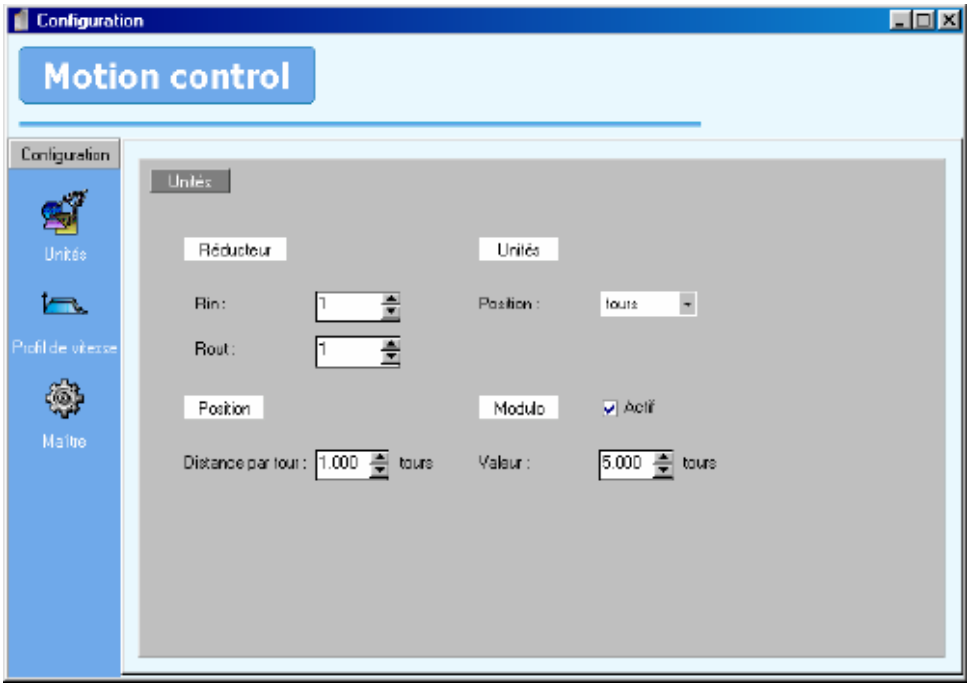
7-2-2- Unité utilisateur

En fonction de l'application, de la mécanique (axe linéaire, rotatif), il est d'intéressant de pouvoir affecter à chaque axe une unité utilisateur représentative : mm, point (point codeur * 4), degrés, radian, pouce, tour, unité quelconque...

En fait, l'unité est utilisée uniquement sur les écrans du DPL afin d'y apporter un confort d'utilisation et de compréhension.

Par exemple, si le choix de l'unité est « mm », dans l'écran de Configuration « Unités » du DPL, la vitesse sera exprimée en mm/s, les accélérations et décélérations en mm/s²...

Cliquer sur **Motion Control \ Configuration \ Unités**, pour paramétrer l'unité de votre axe :



Exemple 1 : Axe infini

Moteur en bout de vis à bille au pas de 5mm. Unités = mm, Rin = 1, Rout = 1, Distance par tour = 5.000, Modulo non activé

Exemple 2 : Axe infini

Moteur avec réducteur de 10. En sortie de réducteur, tourelle 360°, Unités = degrés, Rin = 10, Rout = 1, Distance par tour = 360.000, modulo activé avec une valeur de 360.000

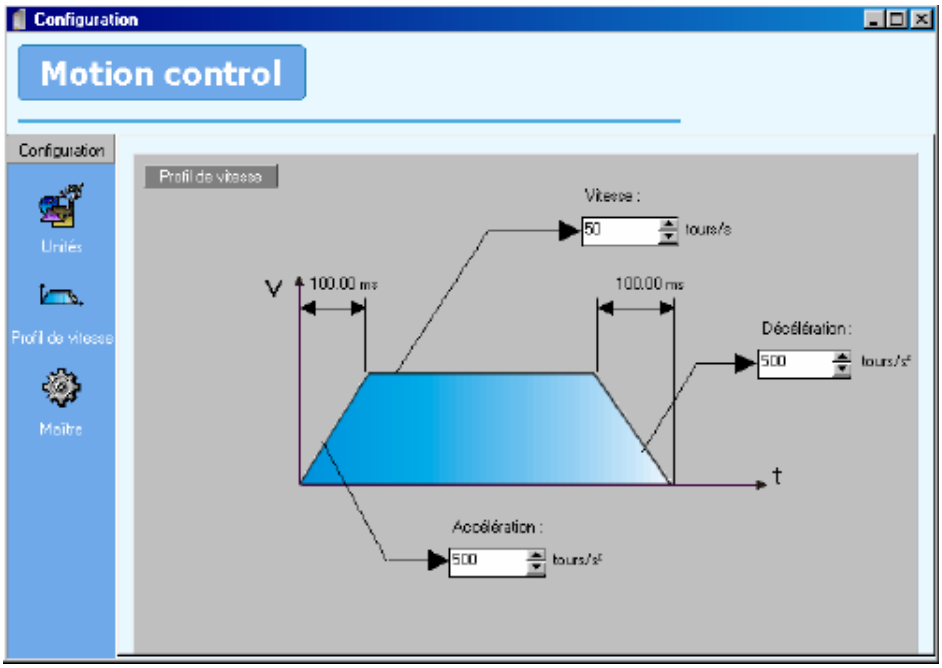
Nota : le nombre de chiffres après la virgule est paramétrable dans le menu *Options / Langage DPL*

7-2-3- Profil de vitesse

Une trajectoire en positionnement intègre les phases d'accélération, de vitesse plateau, de décélération.

Les champs contenus dans la configuration du variateur permettent de donner des valeurs par défaut à ces différentes phases. Les valeurs sont prises en compte à chaque démarrage du variateur, elles sont également utilisées par en mode trajectoire, par les outils de réglage : Motion et Générateur ainsi que par les instructions ACC%, DEC%, VEL%.

Cliquer sur **Motion Control \ Configuration \ Profil de vitesse**,:



La décélération urgente est utilisée pour arrêter le mouvement lorsqu'on utilise les entrées de Fin de course.

7-3- Mode asservi / non asservi

7-3-1- Passage en mode non asservi

L'axe passe en mode non asservi (boucle ouverte) :

- Sur tous défauts.
- Sur erreur de poursuite de l'axe (sauf si l'instruction SECURITY a été affectée).
- A chaque redémarrage du variateur.
- A chaque exécution de l'instruction AXIS OFF à partir d'une tâche.
- Sur un forçage à partir des menus de debug (bouton *enable* en position OFF), du menu communication (arrêt des tâches, redémarrage des tâches, Envoyer les tâches).

L'instruction AXIS_S permet de lire l'état dans lequel se trouve l'axe.

Si une instruction de mouvement est envoyée alors que l'on est en boucle ouverte, elle sera consommée mais le mouvement ne sera pas effectué.

Par exemple :

Tâche Process

```
PROG
...
...             ' le variateur a détecté une erreur de poursuite
...             ' => L'axe passe en mode non asservi
MOVA=1000       ' le mouvement est consommé mais non effectué
OUT (3)=1       ' Activation de la sortie n°3
MOVA=2000       ' le mouvement est consommé mais non effectué
OUT (3)=0       ' Désactivation de la sortie n°3
...             ' La sortie S1 est passée fugitivement à 1 car
...             ' L'instruction Mova(2000)a pris peu de temps au système
END PROG
```

7-3-2- Passage en mode asservi

Pour que l'axe servo puisse piloter et contrôler les mouvements, il est nécessaire de le passer en mode asservi.

L'axe passe en mode asservi (boucle fermée) :

- A chaque exécution de l'instruction AXIS ON à partir d'une tâche.
- Sur un forçage à partir des menus de debug (bouton *enable* en position ON).

L'instruction AXIS_S permet de lire l'état dans lequel se trouve l'axe.

La prise en compte de l'instruction Axis est effectuée au bout d'environ 300 µs. Pour s'assurer que l'asservissement est effectif, écrire :

Axis On

Wait AXIS_S=On

7-4- Prise d'origine

7-4-1- Définition :

La Prise d'origine permet au système de déterminer l'origine mesure de l'axe, celle-ci étant perdue à chaque coupure d'alimentation.

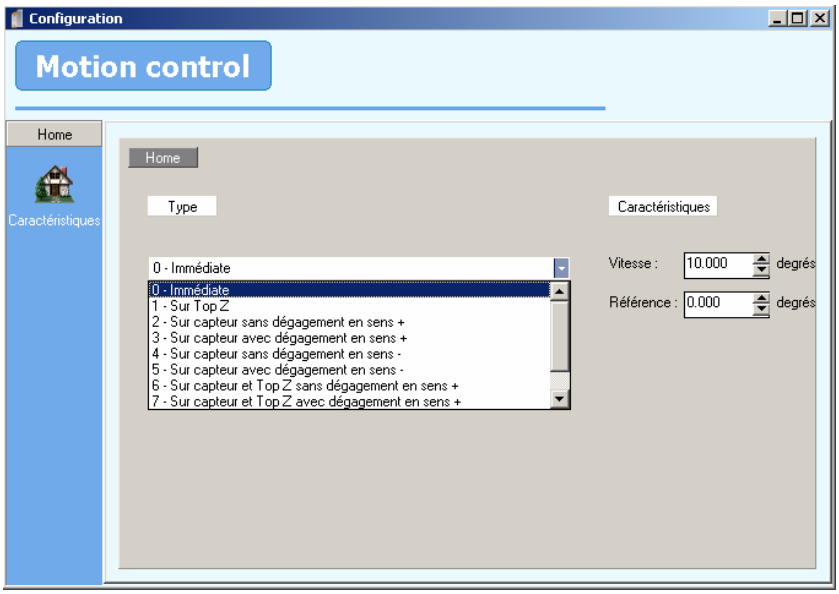
La prise d'origine machine (P.O.M) permet de référencer la position moteur par rapport à une position de la mécanique.

Différents types de POM sont disponibles : immédiat, sur capteur, avec dégagement.

Un cycle de POM force le compteur de position moteur à une valeur de référence.

7-4-2- Configuration de la POM sous DPL :

Pour accéder au paramétrage de la POM, aller dans **Motion control \ Home**


Configuration
Motion control
Home
Caractéristiques
Home
Type
Caractéristiques
0 - Immédiate
0 - Immédiate
1 - Sur Top Z
2 - Sur capteur sans dégagement en sens +
3 - Sur capteur avec dégagement en sens +
4 - Sur capteur sans dégagement en sens -
5 - Sur capteur avec dégagement en sens -
6 - Sur capteur et Top Z sans dégagement en sens +
7 - Sur capteur et Top Z avec dégagement en sens +
Vitesse :
10.000
degrés
Référence :
0.000
degrés

A partir de cet écran, on configure le type de POM, la vitesse et position de référence à charger dans le compteur de position.

Informations :

- Le type choisi dans cet écran est utilisé uniquement sur un mouvement HOME déclaré à partir du tableau Trajectoires lorsque le variateur travaille en mode « trajectoires préenregistrées »
- Si on utilise l'instruction HOME dans une tâche basic, le type doit être indiqué dans l'instruction.

  Exemple : de POM sur top Z -> HOME (1)
- La vitesse de l'axe pendant la POM correspond à la vitesse saisie dans cet écran. Si pendant la POM, l'instruction VEL ou VEL% est exécutée, la vitesse de l'axe est alors modifiée.
- L'instruction Home est bloquante pour la tâche DPL. Si l'on souhaite arrête une POM en cours d'exécution, il faut à partir d'une autre tâche : faire un HALT de la tâche contenant l'instruction HOME, puis un STOP de l'axe.

7-4-3- Les types de POM :

Type 0 : immédiate :

Le compteur de position est forcé à la valeur de référence de façon immédiate.

Exemple : Référence = 100 dans la fenêtre de saisie

HOME (0) ‘ position moteur = 100

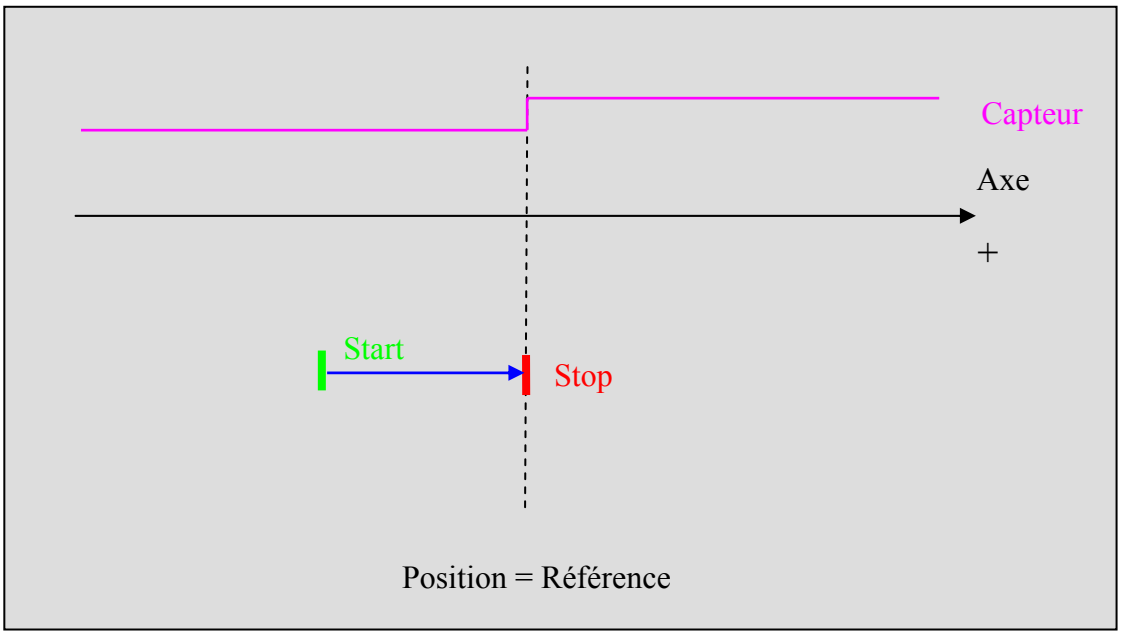
A) Type 1 : sur TOP Z :

Le moteur n'effectue aucun déplacement mais sa position est recalculée par rapport au Top Z moteur et à la valeur de référence. On obtient une position se situant entre +/- ½ tour ou référence +/- ½ tour moteur.

B) Type 2 : Sur capteur, en sens +, sans dégagement

Le variateur lance un mouvement infini en sens + et attend un front montant sur l'entrée HOME.

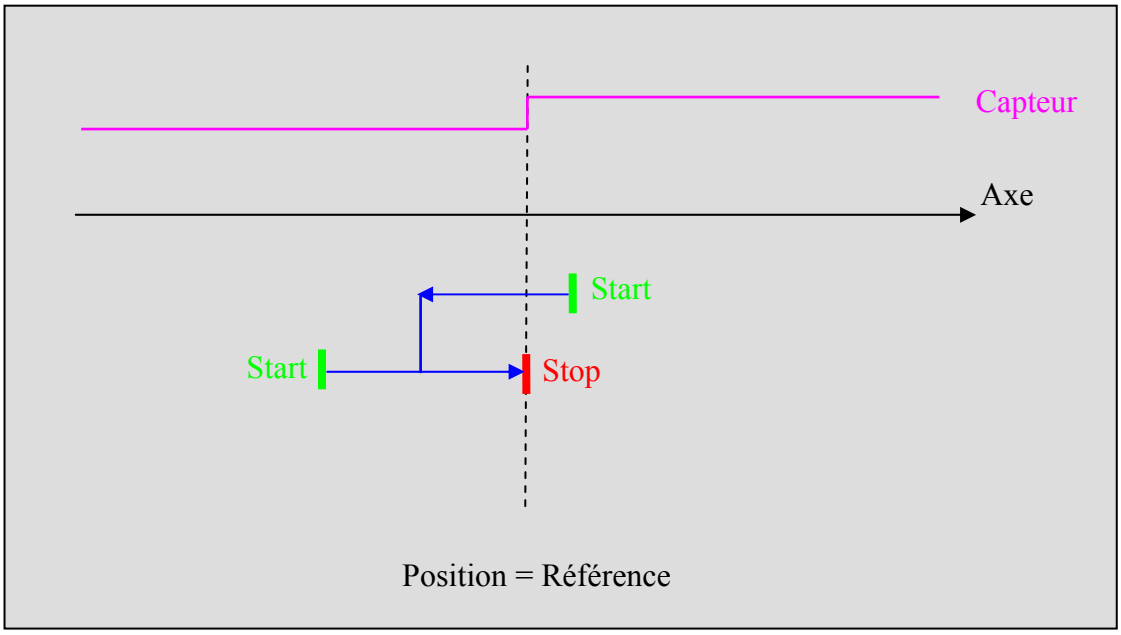
La position est alors forcée à la valeur de référence et le moteur s'arrête sur cette position.



C) Type 3 : Sur capteur, en sens +, avec dégagement

Si l'entrée HOME est déjà à 1 alors le variateur lance en premier un mouvement infini en sens – pour se dégager du capteur HOME.

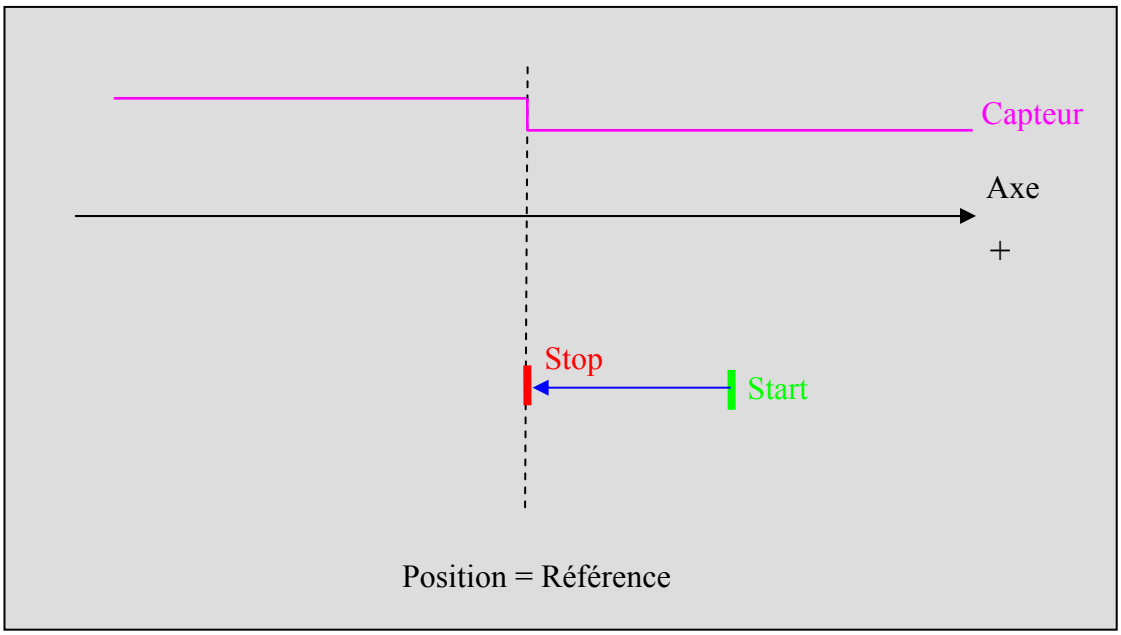
Le variateur lance ensuite un mouvement infini en sens + et attend un front montant sur l'entrée HOME pour forcer la position à la valeur de référence et le moteur s'arrête à cette position.



D) Type 4 : Sur capteur, en sens -, sans dégagement

Le variateur lance un mouvement infini en sens - et attend un front montant sur l'entrée HOME.

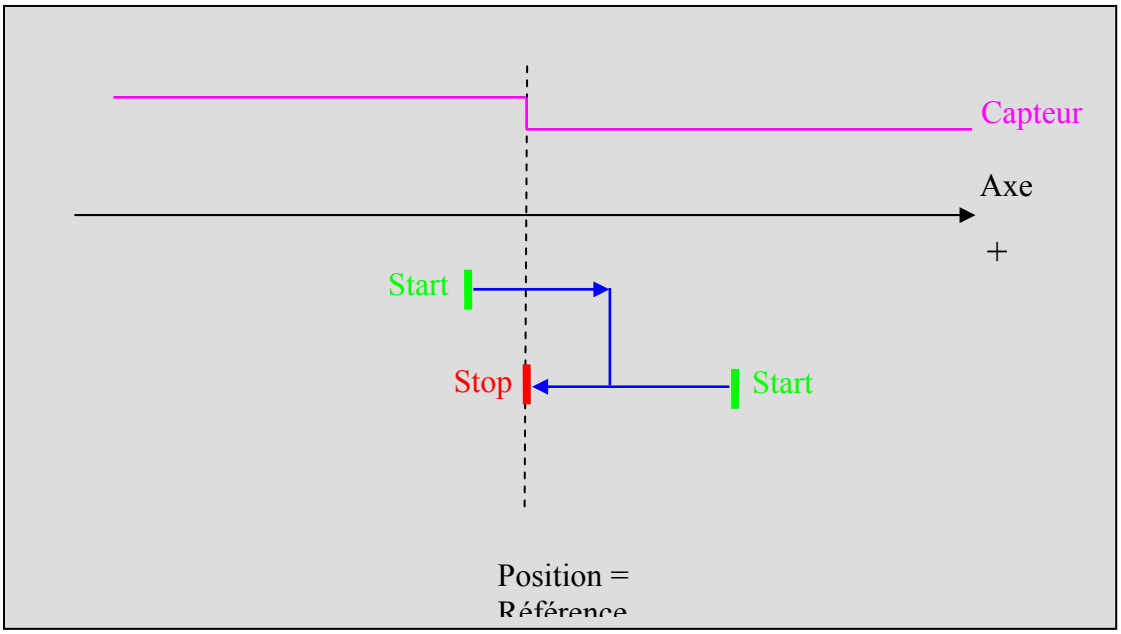
La position est alors forcée à la valeur de référence et le moteur s'arrête sur cette position.



E) Type 5 : Sur capteur, en sens -, avec dégagement

Si l'entrée HOME est déjà à 1 alors le variateur lance en premier un mouvement infini en sens + pour se dégager du capteur HOME.

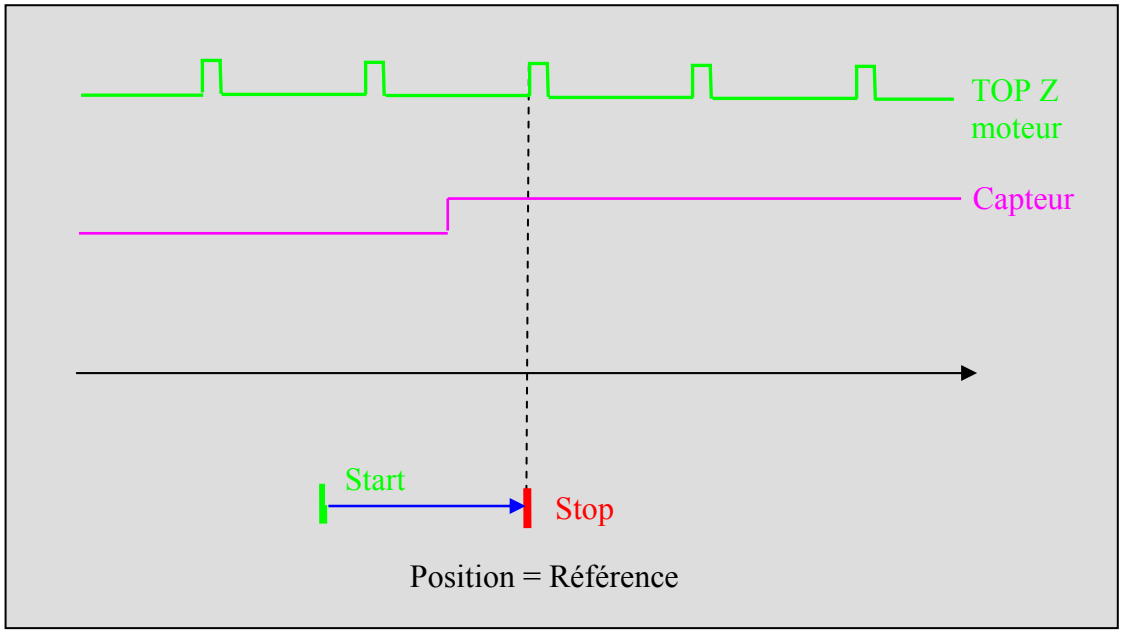
Le variateur lance ensuite un mouvement infini en sens - et attend un front montant sur l'entrée HOME pour forcer la position à la valeur de référence et le moteur s'arrête à cette position.



F) Type 6 : Sur capteur et TOP Z, en sens +, sans dégagement

Le variateur lance un mouvement infini en sens + et attend un front montant sur l'entrée HOME puis le passage par le TOP Z moteur.

La position est alors forcée à la valeur de référence et le moteur s'arrête sur cette position.

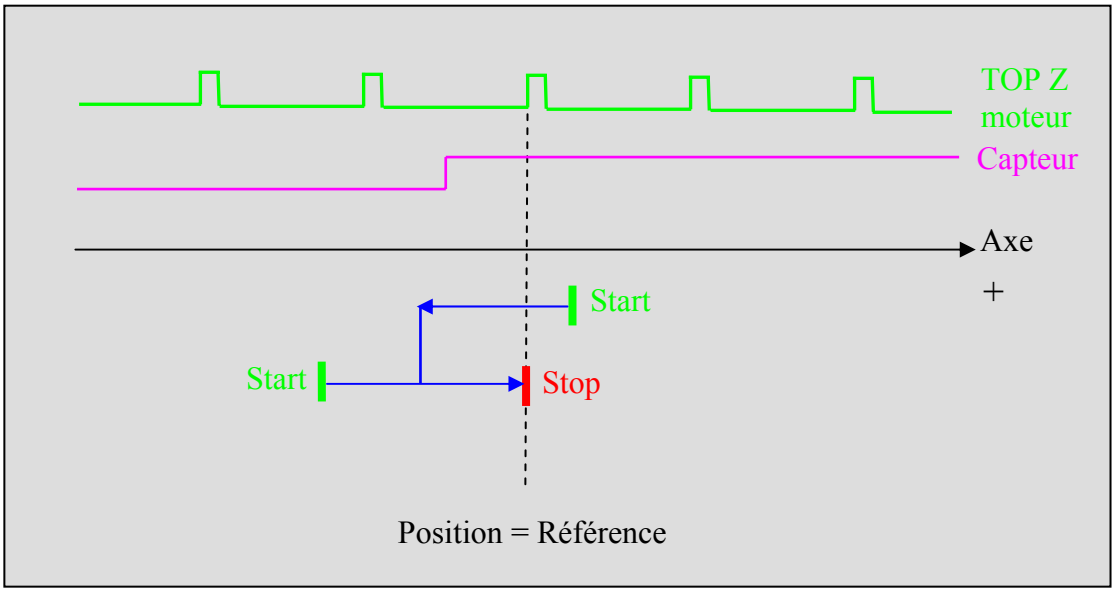


G) Type 7 : Sur capteur et TOP Z, en sens +, avec dégagement

Si l'entrée HOME est déjà à 1 alors le variateur lance en premier un mouvement infini en sens - pour se dégager du capteur HOME.

Le variateur lance ensuite un mouvement infini en sens + et attend un front montant sur l'entrée HOME puis le passage par le TOP Z moteur.

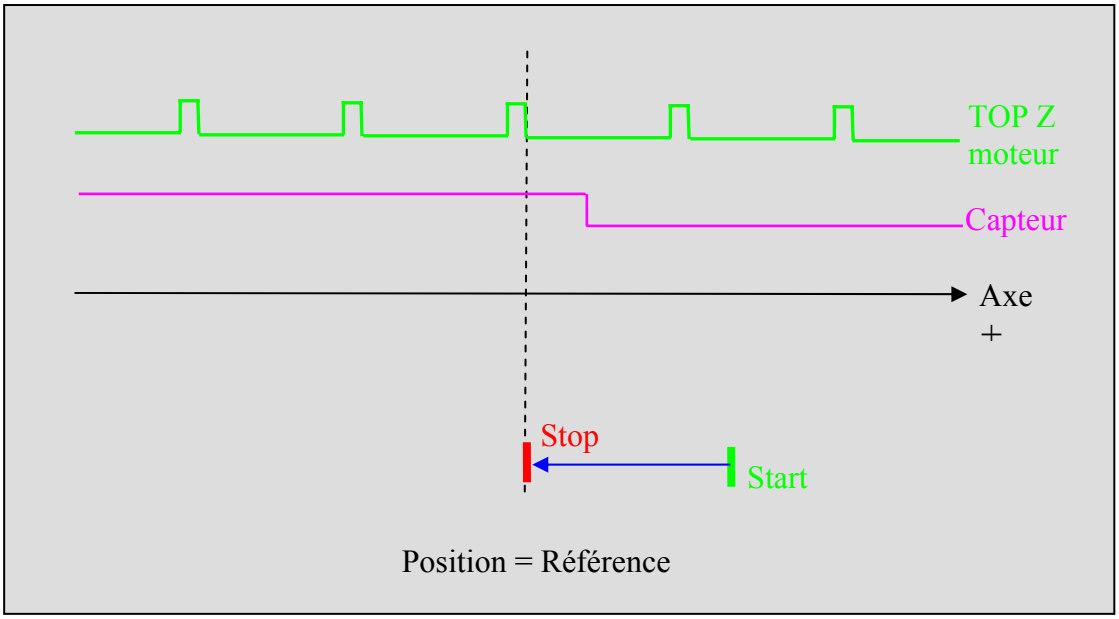
La position est alors forcée à la valeur de référence et le moteur s'arrête sur cette position.



H) Type 8 : Sur capteur et TOP Z, en sens –, sans dégagement

Le variateur lance un mouvement infini en sens - et attend un front montant sur l'entrée HOME puis le passage par le TOP Z moteur.

La position est alors forcée à la valeur de référence et le moteur s'arrête sur cette position.

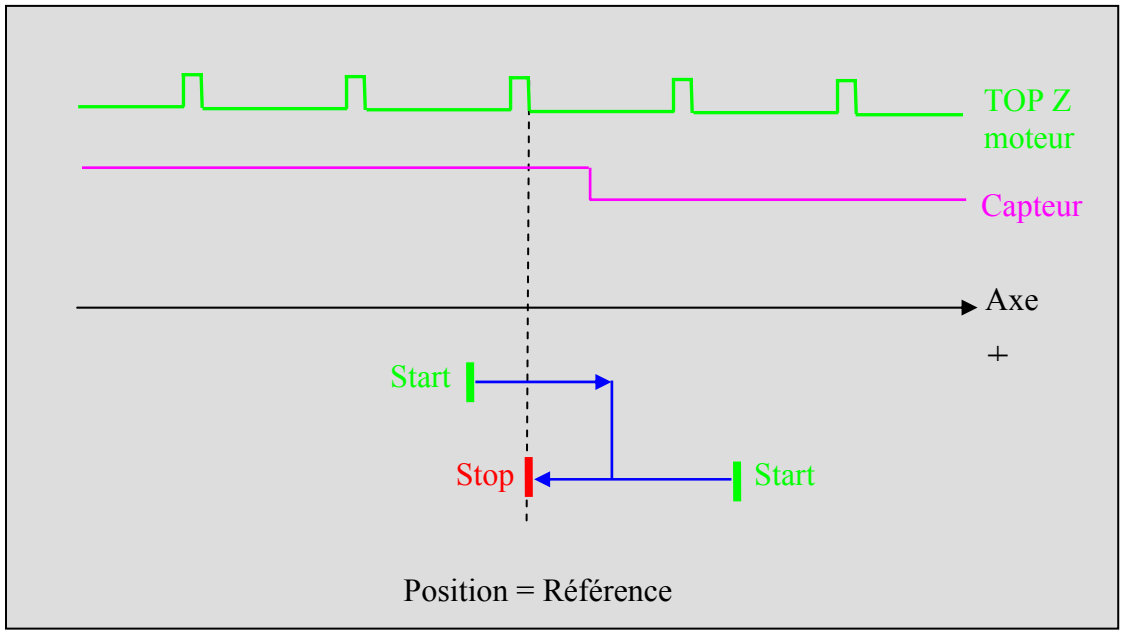


I) Type 9 : Sur capteur et TOP Z, en sens -, avec dégagement

Si l'entrée HOME est déjà à 1 alors le variateur lance en premier un mouvement infini en sens + pour se dégager du capteur HOME.

Le variateur lance ensuite un mouvement infini en sens - et attend un front montant sur l'entrée HOME puis le passage par le TOP Z moteur.

La position est alors forcée à la valeur de référence et le moteur s'arrête sur cette position.



7-5- Déclaration d'un axe en mode virtuel

A partir d'une tâche DPL, il est possible de faire passer un axe en mode virtuel grâce à l'instruction LOOP On. Dans ce mode, le variateur simulera le retour résolveur de façon interne et ainsi tous mouvements seront exécutés virtuellement.

Ce mode est intéressant lors de la phase développement du programme : on peut tester la globalité de l'application sans avoir les variateurs et moteurs connectés.

Dans ce mode, ne pas brancher la puissance sur le connecteur X10

L'instruction LOOP OFF permet de désactiver le mode virtuel.

7-6- Positionnement

7-6-1- Mouvements absolus

a) Départ de mouvement : STTA

Pour lancer un mouvement vers une position absolue et ne pas attendre sa fin pour poursuivre l'exécution de la tâche, on doit utiliser STTA. Cette instruction est très utile si la vitesse ou la position à atteindre doit changer en cours de mouvement. Avec cette fonction, l'erreur absolue est minimale.

Cette instruction est non bloquante pour la tâche (excepté si le buffer de mouvements est plein).

Elle utilise les valeurs courantes d'accélération, de décélération et de vitesse. La syntaxe est :

STTA = Position

Par exemple :

```
VEL%=100
STTA=2000                  ' Départ de l'axe à la position absolue 2000
WAIT POS_S >200            ' Attente position 200
OUT (6)=1                  ' Activation d'une sortie
WAIT POS_S >700            ' Attente position 700
OUT (6)=0                  ' Désactivation de la sortie
WAIT  MOVE_S=0             ' Attente fin de mouvement
```

Dans cet exemple, pendant le mouvement, on peut changer des sorties car la tâche n'est pas bloquée.

Si l'instruction MERGE est activée et que l'on charge plusieurs STTA, les mouvements seront exécutés les uns après les autres sans passer par une vitesse nulle.

Si l'axe est modulo, un lancement à une position sera effectué dans le sens positif si la valeur demandée est positive, sens négatif dans le sens contraire. Par exemple :

Axe modulo 360°

Axe en position initiale à 90°

```
STTA=-10 'déplacement dans le sens - d'une distance de 80°
WAIT MOVE_S=0
STTA=350 'déplacement dans le sens + d'une distance de 340°
WAIT MOVE_S=0
STTA=20  'déplacement dans le sens + d'une distance de 30°
WAIT MOVE_S=0
```

b) Mouvement : MOVA

La fonction MOVA envoie l'axe à une position absolue. Elle utilise les valeurs courantes d'accélération, de décélération et de vitesse. La syntaxe est :

MOVA = Position

Cette fonction envoie l'axe à la position absolue dont la valeur est <Position>. Le programme attend la fin du mouvement avant de continuer. L'erreur de positionnement absolue est minimale.

Par exemple :

```
MOVA=100
CALL Percage
MOVA=0
```

L'instruction MOVA est bloquante pour la tâche tant que le mouvement n'est pas terminé (condition MOVE_S=0).

MOVA=100 est équivalent à STTA =100

WAIT MOVE_S=0

c) Trajectoire : TRAJA

La fonction Trajectoire est conçue pour simplifier la définition de mouvements complexes.

Elle permet de lancer un mouvement vers une position absolue, avec une vitesse spécifique.

Syntaxe de la fonction TRAJ :

TRAJA (<Positon>, <Vitesse>)

Par exemple :

TRAJA (500,2000)

Cet exemple est équivalent à :

```
VEL=2000
STTA = 500
```

Si l'instruction MERGE est activée et que l'on charge plusieurs TRAJA ou TRAJR, les mouvements seront exécutés les uns après les autres sans passer par une vitesse nulle. Par exemple :

MERGE On

TRAJA(500,2000)

TRAJA(1000,50) 'passage en petite vitesse à la position 500

7-6-2- Mouvements relatifs

a) Départ de mouvement : STTR

Pour lancer un mouvement vers une position relative et ne pas attendre sa fin pour poursuivre l'exécution de la tâche, on doit utiliser STTR. Cette instruction est très utile si la vitesse ou la position à atteindre doit changer en cours de mouvement.

Cette instruction est non bloquante pour la tâche (excepté si le buffer de mouvements est plein).

Elle utilise les valeurs courantes d'accélération, de décélération et de vitesse. La syntaxe est :

STTR = Position

Par exemple :

```
VEL%=100                    ' Vitesse rapide
VR1=POS_S
STTR=2000                   ' Départ de l'axe à la position relative 2000
BOUCLE:
VR2=POS_S
VR2=VR2-VR1
IF VR2 < 100 GOTO BOUCLE    ' Attente position +100
VEL%=10                     ' Vitesse lente
WAIT MOVE_S=0               ' Attente fin de mouvement
```

Dans cet exemple, pendant un mouvement, la vitesse peut être modifiée car l'exécution du programme n'est pas bloquée.

Si l'instruction MERGE est activée et que l'on charge plusieurs STTR dans le variateur, les mouvements seront exécutés les uns après les autres sans passer par une vitesse nulle.

b) Mouvement : MOVR

La fonction MOVR envoie l'axe à une position relative. Elle utilise les valeurs courantes d'accélération, de décélération et de vitesse. La syntaxe est :

MOVR (Distance)

Cette fonction envoie l'axe à une valeur relative <Distance>. Le programme attend la fin du mouvement avant de continuer.

Par exemple :

```
VB1=0
BOUClE:
  MOVR=100
  CALL PERCAGE
  VB1=VB1+1
  IF VB1<10 Goto BOUCLE
```

L'instruction MOVR est bloquante pour la tâche tant que le mouvement n'est pas terminé (condition MOVE_S=0).

MOVR=100 est équivalent à STTR =100

WAIT MOVE_S=0

c) Trajectoire : TRAJR

La fonction Trajectoire est conçue pour simplifier la définition de mouvements complexes.

Elle permet de lancer un mouvement vers une position relative, avec une vitesse spécifique.

Syntaxe de la fonction TRAJ :

TRAJR (<Positon>, <Vitesse>)

Par exemple :

TRAJR (500,2000)

Cet exemple est équivalent à :

```
VEL=2000
  STTR = 500
```

Si l'instruction MERGE est activée et que l'on charge plusieurs TRAJA ou TRAJR, les mouvements seront exécutés les uns après les autres sans passer par une vitesse nulle. Par exemple :

MERGE On

TRAJR(500,2000)

TRAJR(1000,50) 'passage en petite vitesse à la position 500 et arrêt du mouvement à la position 1500.

7-6-3- Mouvements infinis

Pour lancer un mouvement continu, il faut utiliser l'instruction STTI. L'axe se déplace alors à sa vitesse courante.

Cette instruction est non bloquante pour la tâche (excepté si le buffer de mouvements est plein).

L'instruction STOP ou SSTOP est nécessaire pour arrêter un mouvement continu. Le sens de déplacement est défini par le caractère "+" ou "-".

Syntaxe :

STTI Signe

Exemple :

```
WAIT  INP(4)=On
STTI+
WAIT  INP(4)=Off
STOP
```

7-6-4- Arrêt d'un mouvement

Pour arrêter un mouvement, il faut utiliser les instructions STOP ou SSTOP. Elles arrêtent l'axe via leur décélération programmée et elles vident le buffer de mouvement.

L'instruction STOP est bloquante pour la tâche tant que le mouvement n'est pas terminé (condition MOVE_S=0) alors que SSTOP n'est pas bloquante.

Syntaxe : STOP

Exemple : déplacement jusqu'à un capteur.

```
STTI(+)
WAIT  INP(4)=On
STOP
```

L'instruction AXIS OFF arrête aussi le mouvement mais sans aucun contrôle car l'asservissement est inhibé.

7-6-5- Positionnement par bus de communication

Il est possible d'exécuter des mouvements via le bus de communication en modifiant directement les paramètres du générateur de mouvement (voir le fichier ..\SERAD\iDpl\Data\Modbus.htm).

A) Profil de vitesse :

- _MOTION_PROJECT_VEL permet de spécifier la vitesse courante en unité par seconde.

- _MOTION_PROJECT_ACC permet de spécifier l'accélération courante en unité par seconde ².
- _MOTION_PROJECT_DEC permet de spécifier la décélération courante en unité par seconde ².
- _MOTION_PROJECT_VELACCDEC permet de spécifier le profil de vitesse en pourcentage des paramètres de l'écran Motion Control \ Profil de vitesse

B) Positionnement :

- _MOTION_PROJECT_HOME permet de démarrer une prise d'origine selon la valeur du paramètre
- _MOTION_PROJECT_STTA permet de démarrer un mouvement absolu à la valeur du paramètre
- _MOTION_PROJECT_STTR permet de démarrer un mouvement relatif à la valeur du paramètre
- _MOTION_PROJECT_SSTOP permet d'arrêter le mouvement en cours.

7-7- Synchronisation

7-7-1- Arbre électrique :

A) GEARBOX :

Cette instruction permet de réaliser un arbre électrique entre un codeur et le moteur (axe esclave).

Syntaxe :

GEARBOX(<Numérateur>, <Dénominateur>, <Réversible>)

<Numérateur> / <Dénominateur> définit le rapport entre un tour du moteur esclave et un tour du codeur. . En fait, pour <Dénominateur> incréments sur le maître, le moteur va bouger de <Numérateur> incréments.

<Réversible> indique si la boite est réversible ou pas, il est mémorisé et sera réutilisé à chaque nouvelle instruction GEARBOXRATIO, STARTGEARBOX

Cette instruction est non bloquante pour la tâche (excepté si le buffer de mouvements est plein).Tant que la liaison entre le maître et l'esclave ne sera pas coupée, l'instruction MOVE_S sera égale à 1 (même si l'esclave est arrêté).

L'instruction GEARBOX initialise de façon interne la valeur de GEARBOXRATIO à 1.

Exemple : Si Numérateur = 1 et Dénominateur = 2, pour 1 tour codeur maître, le moteur esclave tourne de 0.5 tour.

Le Numérateur est de type réel.

Le dénominateur est de type réel.

Un gearbox avec des valeurs <Numérateur> ou <Dénominateur> différentes de 1, affecte la mise à l'échelle de la position du codeur maître (à prendre en compte si l'on traite cette position ou une boîte à contact sur l'axe maître).

B) STARTGEARBOX :

Cette instruction permet de lancer un arbre électrique suivant une distance d'accélération et un rapport défini précédemment par GEARBOX. Le rapport entre le maître et l'esclave est : ratio × <Numérateur> / <Dénominateur>, avec <Numérateur> et <Dénominateur> définis dans l'instruction GEARBOX.

Syntaxe : STARTGEARBOX (<Distance acc. Maître>)

< Distance acc. Maître > est du type réel

Avec Ratio correspondant à la valeur de GEARBOXRATIO.

C) GEARBOXRATIO :

Cette instruction permet de modifier le rapport de réduction d'une liaison arbre électrique.

L'instruction STARTGEARBOX ayant déjà été exécutée.

Syntaxe : GEARBOXRATIO(<Ratio>,<Distance acc. maître>)

<Ratio> de type réel : le rapport de l'arbre est défini par <Ratio> × <Numérateur> / <Dénominateur>. <Numérateur> et <Dénominateur> sont les paramètres de l'instruction GEARBOX.

La distance d'accélération maître <Distance acc. maître> correspond à la distance sur laquelle le maître va accélérer.

L'instruction est non bloquante et permet de changer de ratio sans arrêter l'arbre électrique.

Un GEARBOXRATION n'affecte pas la mise à l'échelle de la position du codeur maître.

L'exécution de l'instruction GEARBOX initialise de façon interne le <Ratio> de GEARBOXRATIO à 1.

D) STOP :

Cette instruction permet d'arrêter un arbre électrique suivant une décélération identique à celle de l'accélération de STARTGEARBOX.

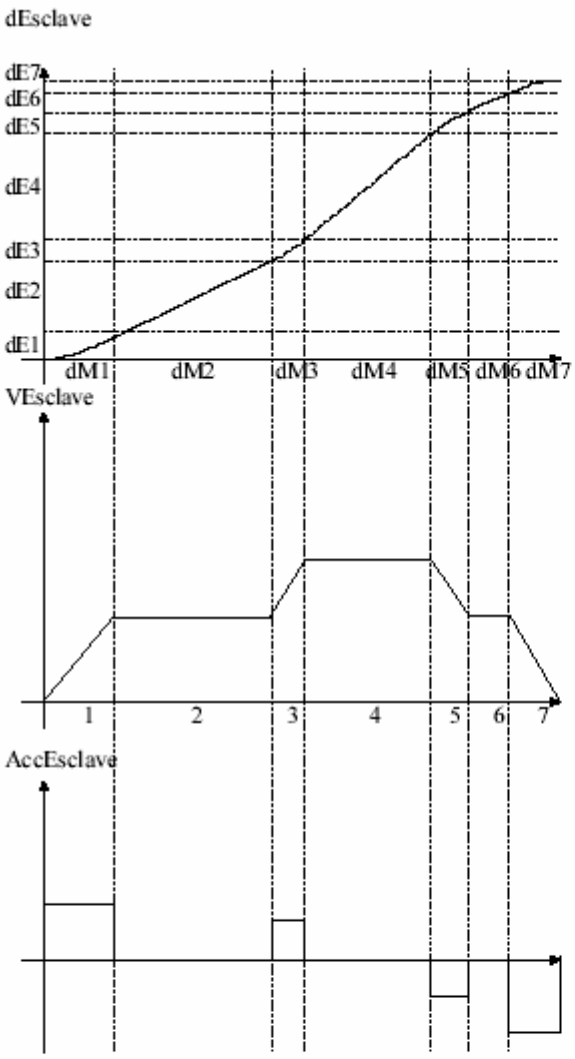
Syntaxe : STOP

E) Exemple :

```
GEARBOX (1, 2, 0)          'Le moteur tourne 2 fois moins vite que le codeur
GEARBOXRATIO (1)           ' la ration finale est de 1* 1 / 2 = 0.5
…
STARTGEARBOX(10)           'Lance l'arbre électrique avec une phase d'accélération
…                          'sur 10 unités du maître
GEARBOXRATIO(2)            ' la ration finale est de 2* 1 / 2 = 1
…
STOP                   'Arrêt de l'arbre électrique avec une phase d'accélération
WAIT MOVE_S=0              'sur 10 unités du maître
```

7-7-2- Mouvements synchronisés

A) Formules générales :



dEa : distance Esclave en phase d'accélération
dMa : distance Maître en phase d'accélération

dEd : distance Esclave en phase de décélération
dMd : distance Maître en phase de décélération

dEs : distance Esclave pendant la phase plateau suivante
dMs : distance Maître pendant la phase plateau suivante

dEp : distance Esclave pendant la phase plateau précédente
dMp : distance Maître pendant la phase plateau précédente

Pour une accélération à partir d'une vitesse nulle :
dEa/dMa =1/2*dEs/dMs

Pour une décélération jusqu'à une vitesse nulle :
dEd/dMd =1/2*dEp/dMp

Pour une phase d'accélération ou de décélération se situe entre deux phases plateaux :

dEa/dMa =1/2*(dEp/dMp+dEs/dMs)

dEd/dMd =1/2*(dEp/dMp+dEs/dMs)

Pour l'exemple on obtient :

dE1/dM1 =1/2*dE2/dM2
dE3/dM3 =1/2*(dE2/dM2+dE4/dM4)
dE5/dM5 =1/2*(dE4/dM4+dE6/dM6)
dE7/dM7 =1/2*dE6/dM6

B) Mouvement : MOVS

L'instruction MOVS permet d'effectuer une synchronisation entre un axe esclave et un maître.

Cette instruction est non bloquante pour la tâche (excepté si le buffer de mouvements est plein).

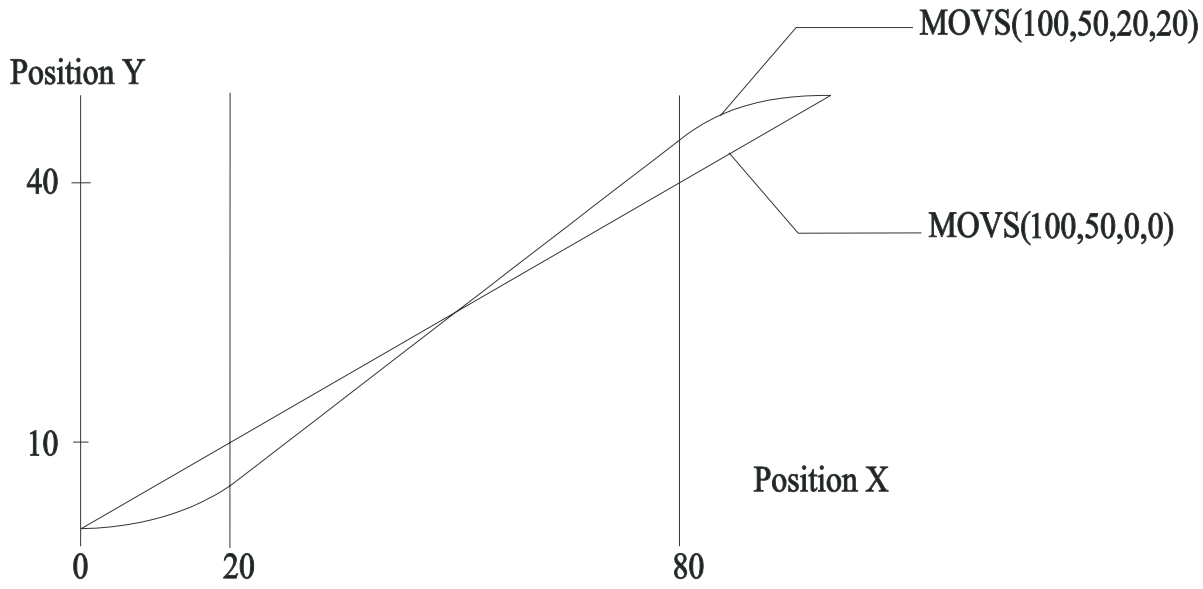
Syntaxe : MOVS (<Dist. maître>, <Dist . esclave>, <Dist. d'accél. maître><Dist. de décél. maître>)

Exemple : MOVS (20, 10, 0, 0) 'pour un déplacement relatif de 20 unités

'sur le maître, l'esclave se déplace de 10

Elle est utilisée pour synchroniser l'axe esclave avec l'axe maître pendant une distance précise de l'axe maître, avec séparément des phases variables d'accélération et de décélération sur l'axe maître.

Par exemple :



Cet exemple représente deux mouvements synchronisés avec et sans les phases d'accélération et de décélération. Quand il n'y a pas de phases d'accélération et décélération, l'axe maître et l'axe esclave doivent avoir la même vitesse pour réduire les phases transitoires. Si les vitesses sont très différentes, les phases d'accélération et de décélération doivent être spécifiées afin d'assurer un comportement mécanique correct.

Les vitesses ne sont pas nécessairement les mêmes et dépendent des phases d'accélération et de décélération. Ceci est causé par la nécessité du respect des distances.

C) Mouvement : STOPS

Lorsque l'axe maître atteint <Pos.maître>, l'axe esclave commencera à décélérer pour atteindre <Pos.esclave>.

Syntaxe : STOPS (<Pos.maître >, <Pos. esclave)

<Pos.maître > du type réel, position dans l'unité du maître.

<Pos.esclave > du type réel, position dans l'unité de l'esclave.

Exemple : STOPS (20, 105) 'Quand l'axe maître aura atteint la position 20,

'l'esclave décélérera pour atteindre la position 105

' sur l'esclave.

Attention : L'appelle de l'instruction STOPS remet le flag STOPS_S à zéro.

D) Etat : STOPS_S

Ne sert que si l'instruction STOPS a été appellé précédemment. Ce flag indique si la position esclave donné par l'instruction STOPS n'a pu être atteinte. Il est remis à zéro après chaque lecture.

Retourne 1 si :

1er cas la position esclave demandé par l'instruction STOPS n'est pas réalisable (ex : la position esclave demandée par STOPS est déjà dépassée.)

2ième cas : si la vitesse esclave est nulle (en phase de plateau).

Sinon retourne 0

Syntaxe : VF0 = STOPS_S

Exemple :
```
MOVS (20, 10, 0, 0)
...
STOPS (20, 105)
WAIT MOVE_S=0
IF STOPS_S=1 GOTO ERRSTOPS
```

E) Applications:

L'instruction MOVS accepte les combinaisons de paramètres suivantes :

- Phase de changement de vitesse

- Phase de changement de vitesse + Phase plateau
- Phase plateau
- Phase plateau + Phase d'arrêt
- Phase d'arrêt
- Phase de changement de vitesse + Phase plateau + Phase d'arrêt

a) Phases de changement de vitesse

(i) Vitesse initiale nulle :

Dans l'exemple précédent la phase 1 est une phase de changement de vitesse avec vitesse initiale nulle.

MOVS (dM1, dE1, dM1, 0)

Le rapport de vitesse atteint ‡ la fin de cette phase est 2*dE1/dM1

(ii) Vitesse initiale non nulle et inférieure à la vitesse finale :

La phase 3 représente ce type de changement de vitesse.

Le rapport de vitesse initial est dE2/dM2 et le rapport final est dE4/dM4 donc : dE3 = dM3 * (dE2/dM2+dE4/dM4) / 2

MOVS (dM3, dE3, dM3, 0)

Le rapport de vitesse moyen pendant cette phase est dE3/dM3 et est supérieur au rapport de vitesse initial; on a donc une phase d'accélération.

(iii) Vitesse initiale non nulle et supérieure à la vitesse finale :

Ce type de changement de vitesse est représenté en phase 5.

Le rapport de vitesse initial est dE4/dM4 et le rapport final est dE6/dM6 donc :

dE5 = dM5 * (dE4/dM4+dE6/dM6) /2

MOVS (dM5, dE5, dM5, 0)

Le rapport de vitesse moyen pendant cette phase est dE5/dM5 et est inférieur au rapport de vitesse initial; on a donc une phase de décélération.

b) Phases de changement de vitesse + Phase plateau

(i) Vitesse initiale nulle :

Dans l'exemple précédent la phase 1 est une phase de changement de vitesse avec vitesse initiale nulle.

dE10=dE1+dE2=1/2*dM1*dE2/dM2+dE2

dM10=dM1+dM2

MOVS (dM10, dE10, dM1, 0)

(ii) Vitesse initiale non nulle et inférieure à la vitesse finale :

La phase 3 représente ce type de changement de vitesse.

Le rapport de vitesse initial est dE2/dM2 et le rapport final est dE4/dM4 donc :

dE30=dE3+dE4=dM3*(dE2/dM2+dE4/dM4)/2+dE4

dM30=dM3+dM4

MOVS (dM30, dE30, dM3, 0)

Le rapport de vitesse moyen pendant la phase 3 est dE3/dM3 et est supérieur au rapport de vitesse initial; on a donc une phase d'accélération.

(iii) Vitesse initiale non nulle et supérieure à la vitesse finale :
Ce type de changement de vitesse est représenté en phase 5.

Le rapport de vitesse initial est dE4/dM4 et le rapport final est dE6/dM6 donc :

dE50=dE5+dE6=dM5*(dE4/dM4+dE6/dM6)/2+dE6

dM50=dM5+dM6

MOVS(dM50,dE50,dM5,0)

Le rapport de vitesse moyen pendant la phase 5 est dE5/dM5 et est inférieur au rapport de vitesse initial; on a donc une phase de décélération.

c) Phase plateau

Les phases 2, 4 et 6 peuvent être décrites directement :

MOVS (dM2, dE2, 0, 0)

MOVS (dM4, dE4, 0, 0)

MOVS (dM6, dE6, 0, 0)

d) Phase plateau + Phase d'arrêt

Dans les phases 6 et 7 on réalise une phase plateau suivi d'une phase d'arrêt.

dE70=dE6+dE7=dE6+1/2*dM7*dE6/dM6

dM70=dM6+dM7

MOVS (dM70, dE70, 0, dM7)

e) Phase d'arrêt

La phase d'arrêt 7 est décrite directement :

MOVS (dM7, dE7, 0, dM7)

Le rapport de vitesse avant cette phase était 2*dE7/dM7

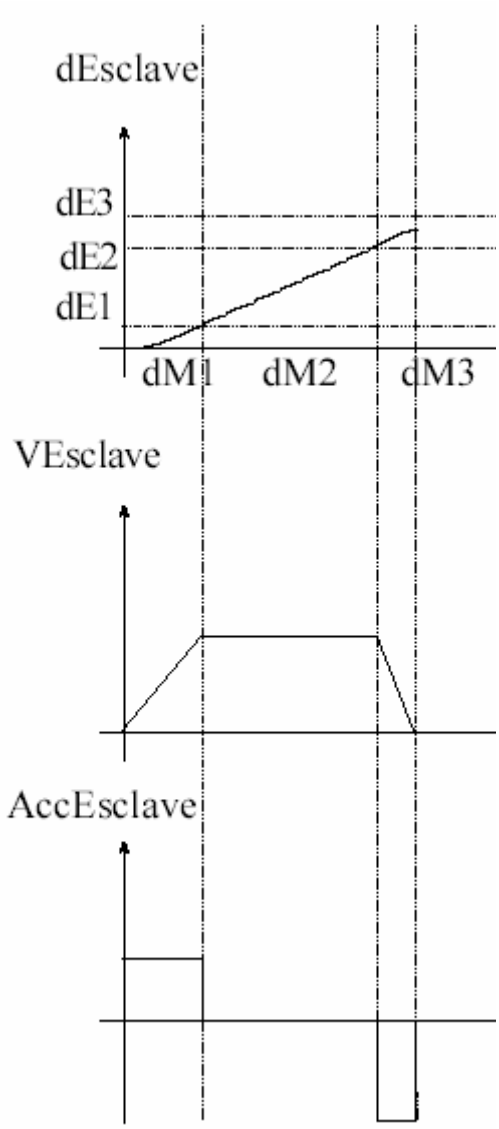
f) Phases de changement de vitesse + Phase plateau + Phase d'arrêt

Dans l'exemple ci-contre on a :

dE1=1/2*dM1*dE2/dM2
dE3=1/2*dM3*dE2/dM2

dE=dE1+dE2+dE3=1/2*(dM1+dM3)*dE2/dM2
dM=dM1+dM2+dM3

MOVS (dM, dE, dM1, dM3)



7-7-3- Mouvement de correction

A) ICORRECTION – fonction de compensation

Syntaxe : ICORRECTION (<Dist.maître>, <Dist.esclave>, <Dist. d'accél maître>)

Unités : <Dist.maître>, <Dist.esclave> : unité utilisateur (Ex : mm, degré,…)

<Dist.d'accél> : unité utilisateur/s²

Types acceptés :<Dist.maître>, <Dist.esclave>, <Dist.d'accél> : réel

Description : Cette fonction permet d'appliquer un mouvement de correction sur un axe esclave pendant une distance de l'axe maître.

Remarques : L'esclave devra au préalable être lié à un maître par une fonction d'arbre électrique (GEARBOX), de mouvement synchronisé (MOVS) avant de lancer une compensation. Au mouvement de synchronisation normal de l'esclave, on superpose le mouvement suivant : Pendant que le maître parcourt une « distance maître », on ajoute un déplacement «distance esclave» avec une accélération et une décélération sur une distance maître de «distance d'accél».

Attention : Toute nouvelle correction est ignorée si une correction est déjà en cours ou si la distance maître est nulle.

B) ICORRECTION_S – Etat de la compensation

Syntaxe : <Variable> = CORRECTION_S

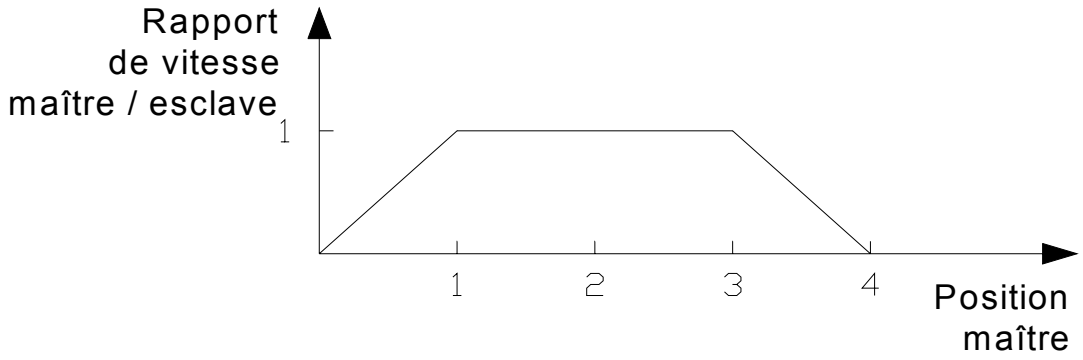
Types acceptés :<Variable> : bit

Description : Cette fonction permet de connaître l'état du cycle de compensation : retourne 1 si ICORRECTION est exécuté sinon renvoie 0.

C) EXEMPLE

1. Mouvement synchronisé :

   MOVS (4, 4, 1, 1)



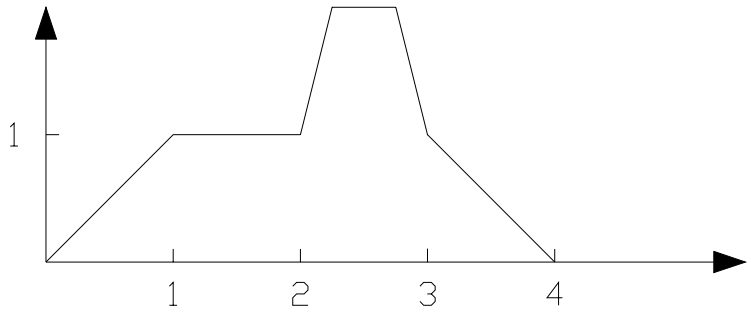
Mouvement synchronisé

2. Mouvement synchronisé + correction :

   MOVS (4, 4, 1, 1)

   WAIT (POSMASTER_S > 2)

   ICORRECTION (1, 1, 0.2)



Mouvement synchronisé + correction

7-7-4- Came

A) Editeur graphique :

L'éditeur de profils de cames accessible à partir du menu **Motion Control \ Editeur de came** facilite la mise en oeuvre.

La came doit être déclaré dans l'écran **Project \ Déclaration \ NomDeVariateur \ came**.

La fonction came permet de réaliser un profil de came sur un axe esclave lié à un axe maître. Ce profil est défini par un tableau de points. Un drive IMD peut stocker jusqu'à 5 cames et 512 points pour l'ensemble des 5 cames.

Chaque point est représenté par une position de l'axe maître et une position de l'axe esclave.

Les valeurs données aux positions de l'axe maître à l'intérieur de la table doivent être des valeurs croissantes.

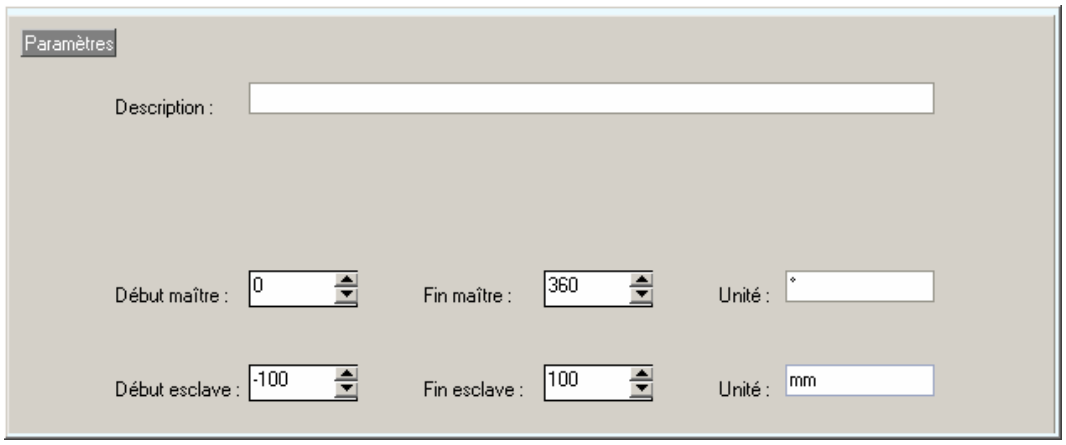
			Début		Fin	
	N°	Mode	Maître	Esclave	Maître	Esclave
▶	0	Ligne	0.000	0.000	0.000	1.000
	1	Manuel	18.947	16.235	28.947	18.235
	2	Manuel	37.895	30.711	47.895	32.711
	3	Manuel	56.842	41.858	66.842	43.858
	4	Manuel	75.789	48.470	85.789	50.470
	5	Manuel	94.737	49.829	104.737	51.829
	6	Manuel	113.684	45.789	123.684	47.789
	7	Manuel	132.632	36.786	142.632	38.786
	8	Manuel	151.579	23.797	161.579	25.797
	9	Manuel	170.526	8.230	180.526	10.230
	10	Auto	189.474	-8.230	189.474	0.000
	11	Auto	208.421	-23.797	208.421	0.000
	12	Auto	227.368	-36.786	227.368	0.000

Un point de came est définie par :

- un mode
- une position maître
- une position esclave
- une tangente maître
- une tangente esclave

La forme de la came dépend du mode de chaque point :

- Ligne : trace une ligne droite qui relie le point courant au point suivant (il y a discontinuité de la vitesse au point actuel puis la vitesse reste constante jusqu'au point suivant).
- Auto : calcule automatique avec un polynôme en x^3 utilisant le point avant, actuel et les deux suivants.
- Manuel : trace une courbe avec un tangente au point actuel de pente tangente maître / tangente esclave.



A partir de la zone «Paramètres» de l'éditeur de came, on peut définir :

↳ Les propriétés de la came : Début de la table (de 0 à 511) et le nombre d'éléments (de 1 à 512) ainsi qu'une description.

↳ Les échelles de l'outils graphique : Début et fin du maître (en X) et début et fin de l'esclave (en Y). Les unités sont à titre indicatif.

Tous les tableaux de cames sont stockés dans le mémoire FRAM du variateur. Pour écrire ou lire un point de came, on utilise les instructions :

<VRx>=ReadCam(<Index>, <Sous index>)

WriteCam(<Index>, <Sous index>)=<VRx>

<Index> de 0 à 511, numéro du point de came en FRAM

<Sous index> de 0 à 3, paramètre du point de came :

↳ 0 pour la position de maître

↳ 1 pour la position de l'esclave

↳ 2 pour la tangente maître

↳ 3 pour la tangente esclave

Le choix du mode de trajectoire pour chaque point dépend des valeurs des différents sous index :

↳ Si Position maître <> Tangente maître alors on a une trajectoire de type Manuel

↳ Si Position maître = Tangente maître et Tangente esclave <> 0 alors on a une trajectoire de type ligne.

↳ Si Position maître = Tangente maître et Tangente esclave = 0 alors on a une trajectoire de type Auto.

B) Came absolue ou relative :

Définie à partir de quel point s'exécute le profil de CAME : en absolue la référence est 0, en relatif la référence est la position des axes au lancement du profil de came.

Exemple :

Profile de came	
Maître	Esclave
0	5
10	7
20	30
30	35
40	30
50	15

- Si la position du maître est 20 et celle de l'esclave est 30, avant le déclenchement de la came, on a les mouvements suivants pour la came absolue :

Came absolue	
Pos. maître	Pos. esclave
20	30
30	35
40	30
50	15

- Si la position du maître est 20 et celle de l'esclave est 30, avant le déclenchement de la came, on a les mouvements suivants pour la came relative :

Came relative	
Pos. maître	Pos. esclave
20	35
30	37
40	60
50	65
60	60
70	45

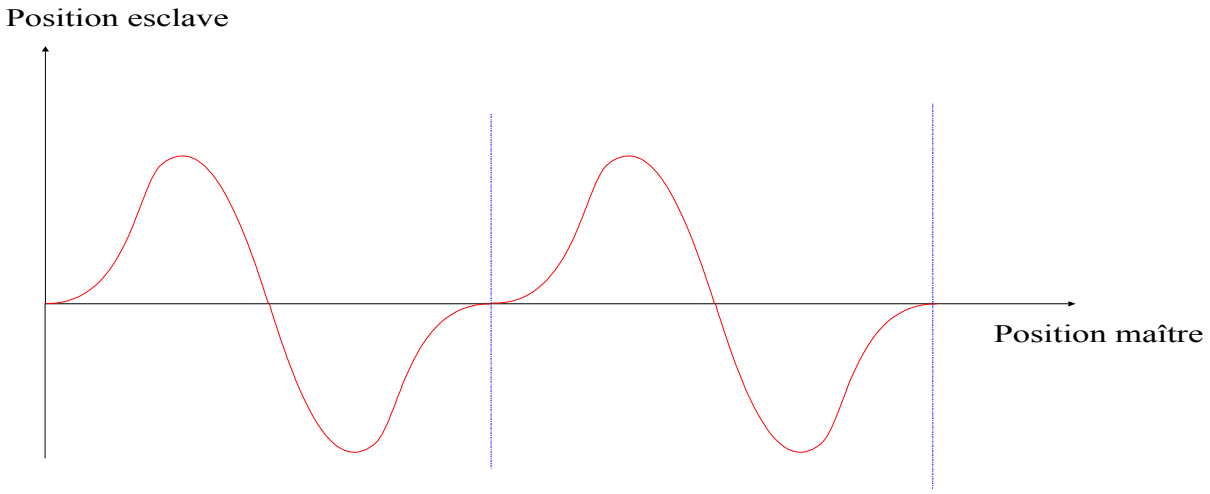
C) Came finie ou came infinie :

Une came mécanique correspond à une came électronique finie. Dans le tableau de points, la 1ère et la dernière valeur de la position de l'esclave sont confondues. Le mouvement de l'esclave sera un mouvement linéaire d'amplitude finie.

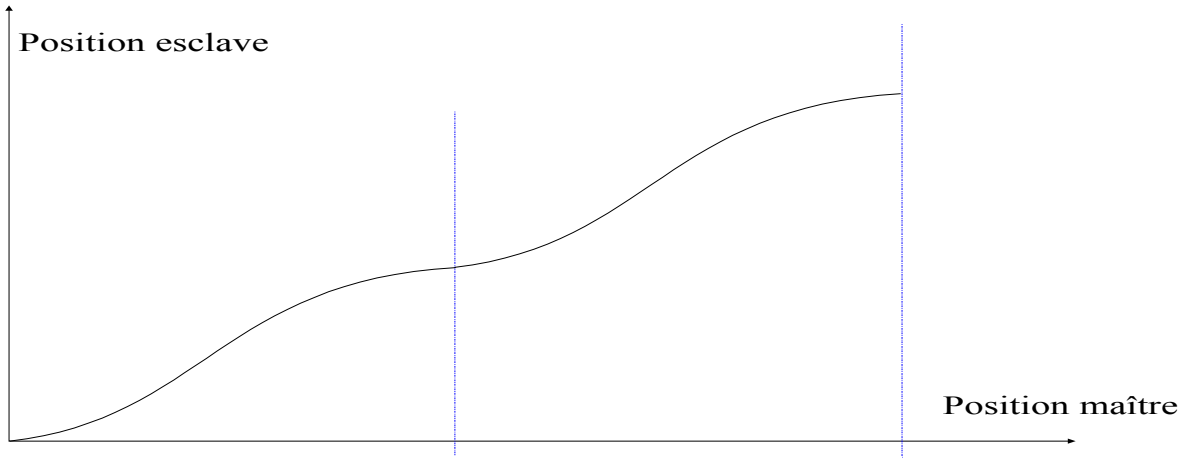
La came électronique permet également de créer un mouvement sur l'axe esclave de type rotatif infini : la position absolue de l'esclave augmente à chaque cycle maître.

Attention : Si l'axe maître ou l'axe esclave est un axe infini, il doit être déclaré en axe modulo à partir du menu **Motion Control** du logiciel iDPL.

Came finie



Came infinie



D) Chargement d'une came :

Pour charger une came, utilisez l'instruction LOADCAM.

Sa syntaxe est la suivante :

LOADCAM(<N°came>, <Absolue>, <Tableau>, <Nombre>, <Mono-coup>, <Réversible>, <Direction>, <GainMaître>, <GainEsclave>, <N°came suivante>, <N°came précédente>)

Description : Cette instruction permet de charger une came dans le variateur.

Limites : <N°came> : numéro de la came (de 1 à 5)

<Absolue> : 1 si came absolue ou 0 si came relative

<Tableau> : indique le 1er point du profile de came (de 0 à 511).

<Nombre> : nombre de points du profil de came (de 2à 512).

<Mono-coup> : Définit le rebouclage automatique de la came.

Rentrez la valeur 0 pour une came qui va se reboucler sur son profil jusqu'à ce qu'un arrêt soit demandé, 1 pour une came qui va exécuter son profil une seule fois.

<Réversible> : Indique si l' <Esclave> doit suivre le <Maitre> dans les deux sens.

↳ Rentrez la valeur 0 pour une came non réversible : si le maître se déplace à l'inverse de son sens normal donné par <Direction>, l'esclave s'arrête ; il repartira lorsque le maître reprendra son sens normal et atteindra la position maître à laquelle l'esclave s'était arrêté.

↳ Rentrez la valeur 1 pour une came réversible : l'esclave suit son profil de came quel que soit le sens d'avance du maître.

<Direction> : Si la came n'est pas réversible, le sens normal de l'avance du maître doit être indiqué. Rentrez la valeur 0 pour un sens indifférent, 1 pour un sens négatif, 2 pour un sens positif.

<GainMaître> : Cœfficient appliqué sur les positions maître du profil de came (valeur réelle à 1 par défaut).

<GainEsclave>: Cœfficient appliqué sur les positions esclave du profil de came (valeur réelle à 1 par défaut).

<N°came suivante> : Mettez 0 si la came ne doit pas être enchaînée sur une autre came. Dans le cas contraire, rentrez le numéro de la came suivante compris entre 1 et 5.

<N°came précédente> : Mettez 0 si la came n'enchaînera pas sur une came précédente. Dans le cas contraire, rentrez le numéro de la came précédente compris entre 1 et 5.

E) Lancement d'une came :

Pour lancer l'exécution d'une came, utilisez l'instruction STARTCAM.

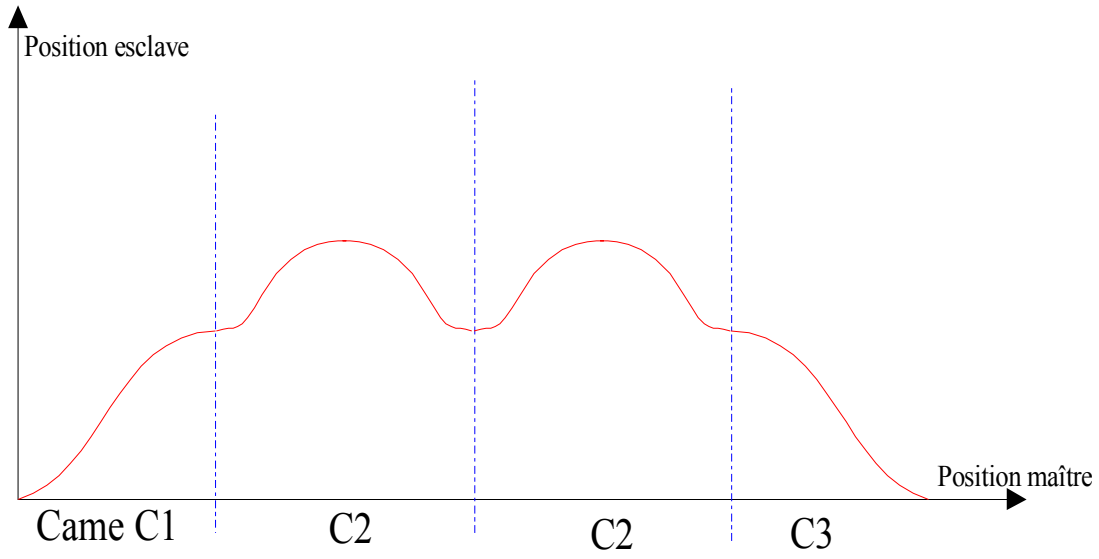
Sa syntaxe est la suivante : STARTCAM(<N°came>)

<N°came> : numéro de la came (de 1 à 5).

F) Enchaînement de cames :

Soit un cycle composé de 3 cames : une came C1 de « profil d'entrée » mono-coup, une came C2 de « profil répétitif »non mono-coup, une came C3 de « profil de sortie » mono-coup.

La came C1 est enchaînée à C2 et C2 est enchaînée à C3.



PROG

.....

‘ Chargement de la came n°1 : 10 points, mono-coup, enchaînée sur came n°2

```
LOADCAM(1,0,0,10,1,1,0,1,1,2,0)
' Chargement de la came n°2  : 76 points, non mono-coup, enchaînée sur came n°3
LOADCAM(2,0,10,76,0,1,0,1,1,3,1)
' Chargement de la came n°3  : 6 points, mono-coup
LOADCAM(3,0,86,6,1,1,0,1,1,0,0)
' Lancement de la came n°1 => exécution de la came1 puis de la came2
STARTCAM(1)
WAIT CAMNUM_S=     2              ' Attente exécution de la came2
.....
WAIT INP(InfoStop)                ' Attente demande d'arrêt
ENDCAM                            ' Arrêt came2 en fin profil
                                  ' et enchaînement sur came3
WAIT MOVE_S=0                     ' Attente came 3 terminée
.....
END PROG
```

G) Etat de la came :

Trois fonctions permettent de connaître l'état courant d'une came.

↳ Instruction MOVE_S : permet de savoir si une came de la carte est en cours d'exécution.

Exemple :

```
IF NOT MOVE_S THEN GOTO FINCAME          'Came arrêtée
IF MOVE _S=1 THEN GOTO CAME_EN_COURS     'Came Lancée
```

↳ Instruction CAMNUM_S : permet de savoir quel numéro de came est en cours d'exécution. La valeur retournée est significative que si MOVE_S est à 1.

Exemple :

```
IF CAMNUM_S=1 THEN GOTO ATTENTE_FIN_CAME_1    'Came 1 en cours
IF CAMNUM_S=2 THEN GOTO ATTENTE_FIN_CAME_2    'Came 2 en cours
```

↳ Instruction CAMSEG_S : permet de savoir quelle numéro d'équation de la came est en cours d'exécution. La valeur retournée est significative que si MOVE _S est à 1.

Exemple :

```
IF CAMSEG_S=1 THEN GOTO ATTENTE_FIN_SEGMENT_1    'Came entre le point 1 et le point 2
IF CAMSEG_S=2 THEN GOTO ATTENTE_FIN_SEGMENT_2    'Came entre le point 2 et le point 3
```

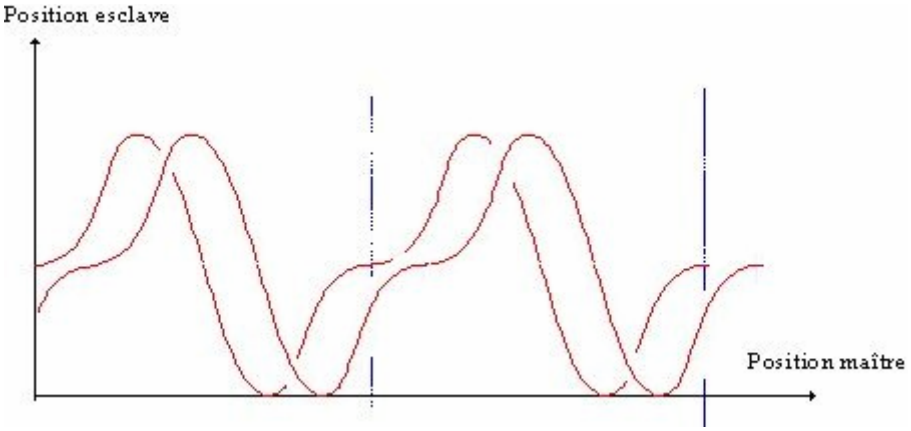
H) Arrêt de la came :

La fonction ENDCAM permet d'arrêter le mouvement de l'esclave à la fin du profil de la came tandis que la fonction STOP met fin au mouvement immédiatement. La syntaxe de l'instruction ENDCAM est la suivante : ENDCAM

Attention : Si ENDCAM s'applique à une came qui a été déclarée en mode non mono-coup et enchaînée avec une autre, la came termine son profil et enchaîne sur la suivante.

I) Déphasage dynamique :

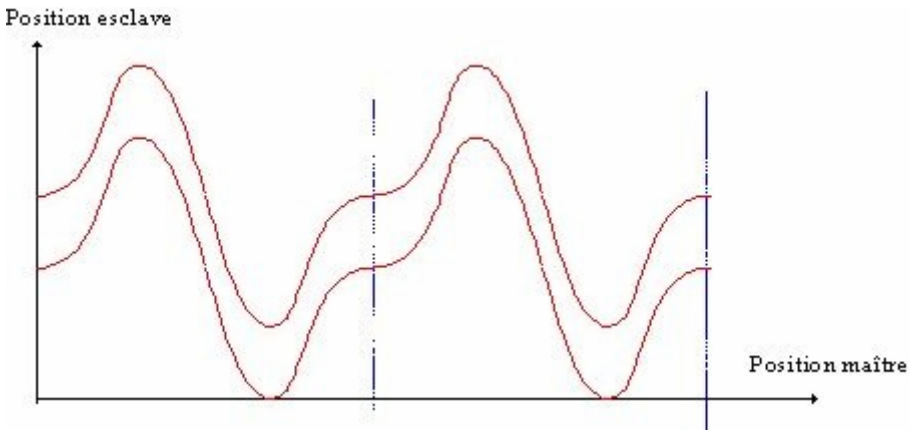
a) Décalage du maître



Le décalage du maître a pour effet de déphaser le cycle du maître par rapport à celui de l'esclave. Dans le cas d'une came rebouclée il est nécessaire de prendre en compte ce décalage pour positionner l'esclave par rapport au maître. Le décalage du maître peut se faire progressivement par l'application d'un paramètre d'accélération. Le décalage est appliqué directement si le mouvement synchro n'est pas en cours ou si l'axe n'est pas embrayé.

```
' Application des offsets immédiatement
MasterOffset(OffsetMaitre,1000)
SlaveOffset(OffsetEsclave,1000)
' Lancement de la came
StartCam(1)
…
…
OffsetMaster= OffsetMaster+10   ' Changement d'offset en cours de cycle
MasterOffset (OffsetMaster, 0.1)
```

b) Décalage de l'esclave



Le décalage de l'esclave a pour effet de décaler les positions de l'esclave mais conserve la phase avec le cycle du maître. Il est nécessaire dans tous les cas de prendre en compte ce décalage pour positionner l'esclave par rapport au maître. Le décalage de l'esclave peut se faire progressivement par l'application d'un paramètre d'accélération. Le décalage est appliqué directement si le mouvement synchro n'est pas en cours ou si l'axe n'est pas embrayé.

```
' Application des offsets immédiatement
MasterOffset(OffsetMaitre,1000)
SlaveOffset(OffsetEsclave,1000)
' Lancement de la came
StartCam(1)
…
…
OffsetEsclave= OffsetEsclave+10  ‘ Changement d’offset en cours de cycle
SlaveOffset (OffsetEsclave, 0.1)
```

J) Modification de points d'une came : LOADCAMPOINT

Permet de modifier un point d'une came à partir d'un point en FRAM. Sa syntaxe est la suivante :

LOADCAMPOINT (<N° came>, <N° point>, <Index en FRAM>)

<N° came> : Numéro de la came cible chargée précédemment (de 1 à 5).

<N° point> : Numéro du point cible de la came (de 1 à nb de points de la came).

<Index en FRAM> : Adresse du point source FRAM (de 0 à 511) à envoyer dans le point cible de la came.

Attention : Cette instruction est bloquante pour la tâche (le chargement d'un nouveau point ne peut se faire si la came se trouve entre les 2 polynômes avant et après le point cible). Cette instruction provoque une erreur DPL E11 si la came cible n'a pas été chargée précédemment.

K) Position de l'esclave dans la came : CAMREADPOINT

Syntaxe : <Position Esclave> = CAMREADPOINT (<Position Maître>, <N° came>)

Types acceptés : <Position Esclave> type réel dans l'unité de l'esclave

<Position Maître> type réel dans l'unité du maître

<N° came> : Numéro de la came cible chargée précédemment (de 1 à 5).

Description : Cette instruction permet de calculer la position de l'esclave <Position Esclave> dans la came, correspondant à la position du maître <Position Maître>.

Remarque : Retourne 0 si on n'est pas dans la came sélectionnée.

L) Came déclenchée sur entrée capture :

Il est possible d'exécuter des cames synchronisées en utilisant les fonctions TRIGGER

M) Mise en garde :

- Les valeurs données aux positions de l'axe maître à l'intérieur de la table doivent être des valeurs croissantes.
- Cet écart ne doit pas être trop petit. Il est préférable que le système passe par tous les points successifs, même à vitesse maximale (période d'échantillonnage de 150µs).
- Les cames et les données sauvegardées sont stockées en mémoire FRAM, attention de ne pas utiliser les mêmes adresses.

7-7-5- Multiaxes par CANopen

Il est possible de synchroniser plusieurs drives par échange de position par bus CANopen :

a) Tache du drive d'émission :

```
Prog
  StartCANSendPosition(1,1,210h,10)

Bcl:
  Goto Bcl

EndProg
```

b) Tache de drive de réception :

```
Prog
  StartCANReceivePosition(1,210h,0,20)
```

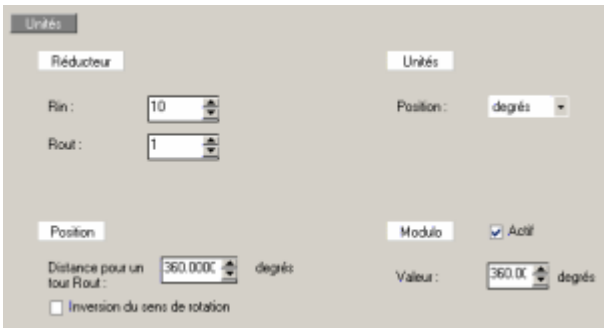
```
  Axis On
  Wait(Axis_S) = On
  Filtermaster 1
  Gearbox(1,1,1)
  Startgearbox(1)

Blc:
  vi0=canposstatus
  If (vi0=2) then
        vi1=vi1+1
        canpostimeoutraz
  Endif
  Goto test

EndProg
```

c) Attention :

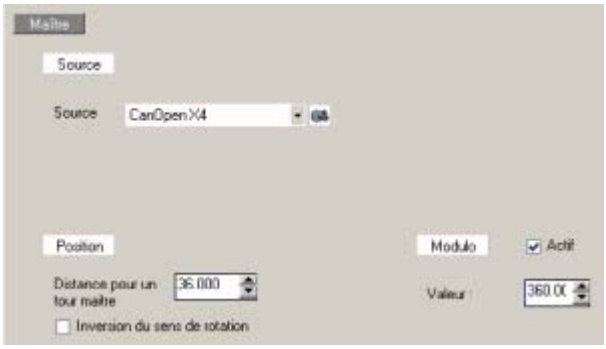
Dans **Motion control \ Maître Esclave**, la source du maître doit être configuré en CANopen X4

Si le maître travaille en modulo, il est impératif d'avoir les mêmes réglages dans l'écran Motion Control \ Unités du maître et l'écran Motion Control \ Maître de l'esclave :

Ex :Maitre



Esclave



Les 2 modulo doivent être identiques et la distance pour 1 tour maître=distance pour 1 Rout * Rout / Rin

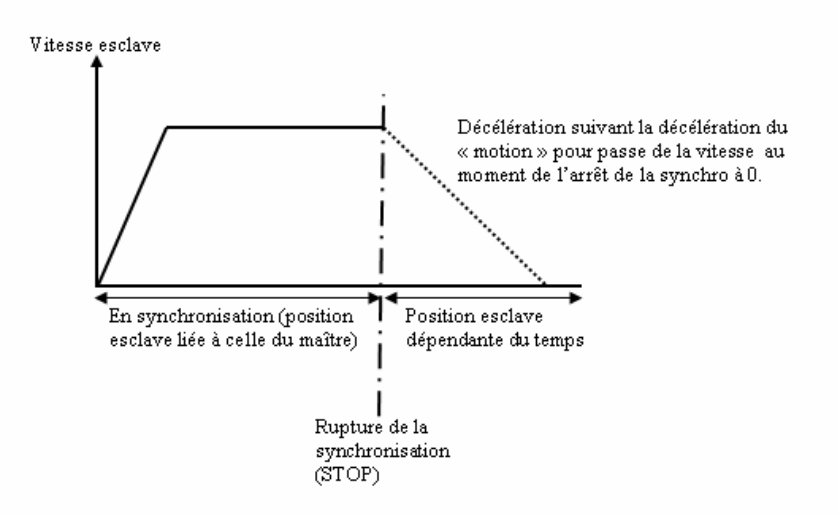
La liste des instructions CAN pour la synchronisation est décrite dans le chapitre Annexe \ CANopen \ liste des instruction.

7-7-6- Débrayage d'un mouvement synchronisé

Un mouvement synchronisé peut être terminer par :

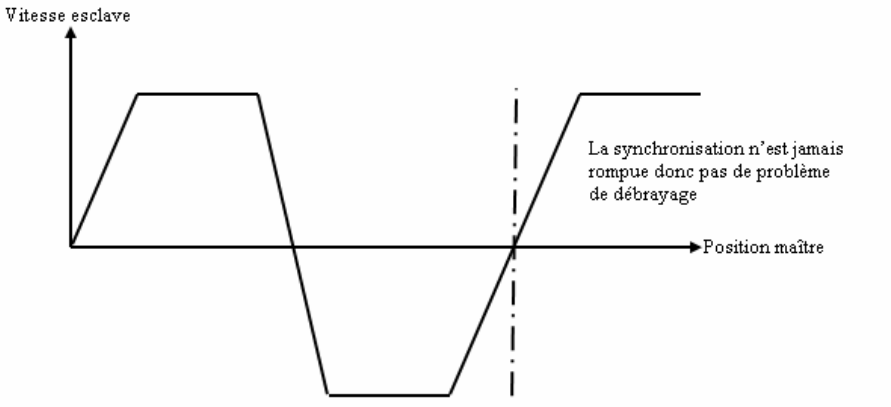
- un STOP : fin de mouvement immédiat
- un STOPS : fin de mouvement sur condition maître et esclave
- un ENDCAM : arrêt en fin de came pour une came
- une fin de mouvement « naturelle » (ex fin d'une came)

Dans tous les cas au moment de l'arrêt de la synchronisation (débrayage), une rampe de décélération est effectué sur l'esclave :

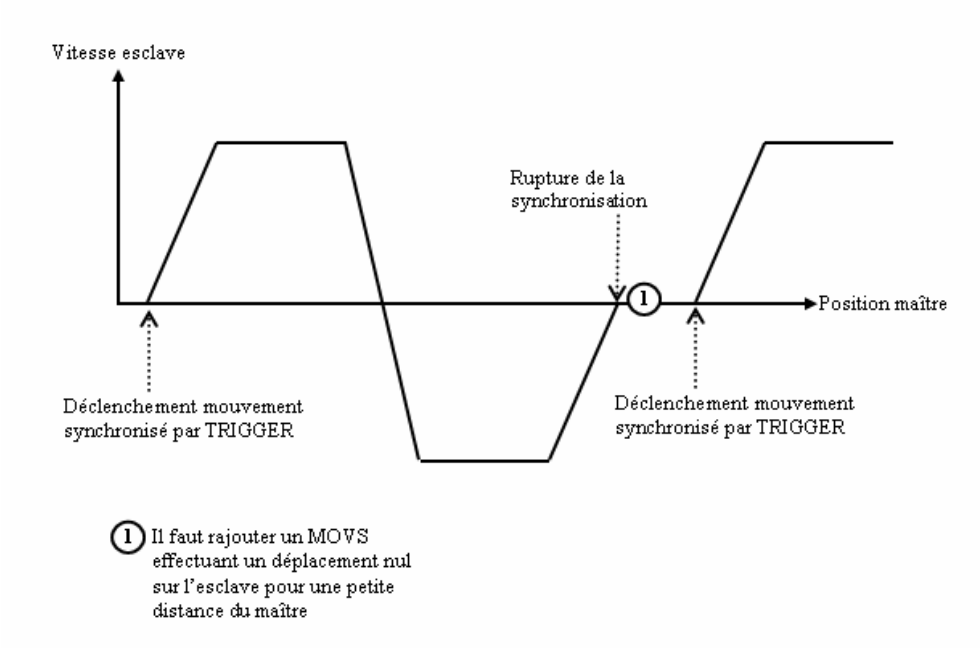

Vitesse esclave
Décélération suivant la décélération du « motion » pour passe de la vitesse au moment de l'arrêt de la synchro à 0.
En synchronisation (position esclave liée à celle du maître)
Position esclave dépendante du temps
Rupture de la synchronisation (STOP)

En pratique il y aura toujours une phase de décélération (même très faible).

Cas d'un cycle continue :



Mouvement synchronisé déclenché :


Vitesse esclave
Rupture de la synchronisation
1
Position maître
Déclenchement mouvement synchronisé par TRIGGER
Déclenchement mouvement synchronisé par TRIGGER
1 Il faut rajouter un MOVS effectuant un déplacement nul sur l'esclave pour une petite distance du maître

7-8- Capture

7-8-1- Capture :

La capture permet d'enregistrer la position courant de l'axe sur un front d'une entrée variateur.

Temps de capture :

	Entrée normale	Entrée rapide
Filtrage	Filtrage	600 µs
Sans filtrage	150 µs	1 µs

A) CAPTURE1 ou CAPTURE2 :

Les instructions CAPTURE1 et CAPTURE 2 sont utilisées pour enregistrer la position courante de l'axe ou la position du maître.

Avec cette instruction, le variateur attend un front sur l'entrée capture. Quand le front est détecté, la position est stockée dans la variable REGPOS1_S. Le flag REG1_S est alors positionné à vrai.

Syntaxe : CAPTURE1 (<Source>, <N° de l'entrée>, <Front>, < Fenêtre >, <Mini>, <Maxi>, <Intérieur>)

<Source> 0 pour position moteur, 1 pour position maître.

<N° de l'entrée> numéro l'entrée sur laquelle on attend le front montant (de 1 à 16).

<Front> 1 sur front montant ou 0 sur front descendant.

<Fenêtre> est vraie alors l'entrée n'est testée que lorsque l'axe est entre les positions <Mini> et <Maxi>.

<Mini> doit toujours être inférieur à <Maxi>.

<Intérieur> permet de définir si le test s'effectue à l'intérieur ou à l'extérieur des bornes <Mini> et <Maxi>

Attention : CAPTURE doit être relancée pour chaque nouvelle capture.

Il est interdit d'utiliser simultanément la même entrée et le même front sur les fonctions de mouvement déclenché, les captures et les compteurs.

B) REG1_S ou REG2_S :

Syntaxe : <VFx>=REG1_S

Description : Cette fonction indique si une capture de position a été effectuée.

Remarques : La valeur retournée n'est vraie qu'une fois par capture. REG1_S est remis automatiquement à 0 sur une opération de lecture. Sur une relance d'une autre capture et si REG1_S vaut 1 alors REG1_S est mis à 0.

C) REGPOS1_S ou REGPOS2_S :

Syntaxe : <Expression>=REGPOS1_S

Types acceptés : Expression : réel

Description : Cette fonction retourne la dernière position capturée sur l'axe par l'exécution de l'instruction CAPTURE1.

D) Exemple :

```
DEMARRECAPTURE :
  CAPTURE1 (0,4,1,On,10,20,On)        'Capture position sur front montant de l'entrée 4,
  ...                                 ' lorsque l'axe du moteur est entre 10 et 20.
ATTENTE :
IF REG1_S = ON THEN                   'Attente d'une capture
  VR1 = REGPOS1_S                     'VR1 = valeur de la position lors de la capture
  GOTO DEMARRECAPTURE
ENDIF
...
GOTO ATTENTE
```

7-8-2- Recalage automatique

A) ENABLERECALE – Fonction de recalage automatique sur capture

Syntaxe : ENABLERECALE (<n° capture>, <Position initiale>, <Accélération>)

Limites : <Position Initiale> : entre 0 et le modulo de l'axe

Types acceptés :<Position Initiale> : Réel

<Accélération> : Réel

Description : Cette instruction recale automatiquement un axe sur un capteur

Remarques : Le paramétrage de la fonction de recalage utilise les paramètres de l'instruction CAPTURE :

<Source> 0 pour position moteur, 1 pour position maître.

<N° de l'entrée> numéro l'entrée sur laquelle on attend le front montant (de 1 à 16).

<Front> 1 sur front montant ou 0 sur front descendant.

<Fenêtre> est vraie alors l'entrée n'est testée que lorsque l'axe est entre les positions <Mini> et <Maxi>.

<Interieur> permet de définir si le test s'effectue à l'intérieur ou à l'extérieur des bornes <Mini> et <Maxi>

<Mini> doit toujours être inférieur à <Maxi>.

L'appel ENABLERECALE annule la fonction CAPTURE dont les paramètres ont été utilisés.

<Position Initiale> indique la position à laquelle se trouve théoriquement le capteur et que l'on mettra dans le compteur. Pour une remise à zéro on indique 0.

<Accélération> indique l'accélération à utiliser pour le recalage

Exemple : …

```
CAPTURE1 (0, 2, 1, 0, 0, 0, 0)      'Capture sur l'esclave et
                                    ' sur front montant de l'entrée 2
ENABLERECALE (1, 0, 1000)           'Utilisation des paramètres
                                    ' de la capture1
                                    ' avec remise de la position à 0
                                    ' et avec une accélération de 1000
…
DISABLERECALE (0)
```

B) DISABLERECALE – Désactivation du recalage

Syntaxe : DISABLERECALE (<Axe>)

Limites : <Axe> : 0 pour l'axe esclave et 1 pour le maître.

Description : Cette instruction annule le recalage automatique d'un axe sur un capteur

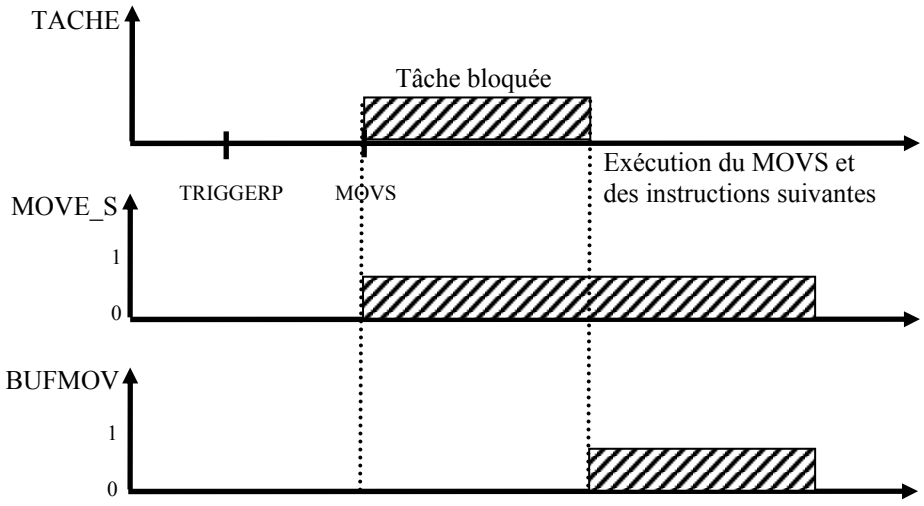
7-9- Mouvements déclenchés

7-9-1- Mouvements déclenchés

Permet de lancer un mouvement sur un évènement :

- une position du maître
- un changement d'état d'une entrée
- une capture

Sur un mouvement déclenché, la tâche qui lance le mouvement déclenché est bloquée jusqu'à déclenchement du mouvement :



A) Instruction : TRIGGERP

Cette instruction indique que le prochain mouvement sera déclenché sur le passage du maître à une position.

Syntaxe : TRIGGERP (<Pos. Maître>, <Sens>)

<Pos.maître > du type réel, position dans l'unité du maître.

<Sens > 0 indifférent, 1 sens négatif, 2 sens positif.

Exemple : STTA =50 'mouvement absolu en 50 sans attente de fin de mouvement

```
…
TRIGGERP (200,2)
STTA =300     ‘mouvement absolu en 300
              ‘ déclenché lorsque la position du maître atteindra 200
              ‘ dans le sens positif
```

B) Instruction: TRIGGERI

Cette instruction indique que le prochain mouvement sera déclenché sur changement d’état d’une entrée.

Syntaxe : TRIGGERI (<N° d’entrée>, <Front>)

< N° d’entrée > de 1 à 16.

< Front > 0 front descendant, 1 front montant.

Exemple :

```
STTA =50 ‘mouvement absolu en 50 sans attente de fin de mouvement
…
TRIGGERI (7,1)
STTA =300     ‘mouvement absolu en 300
              ‘ déclenché sur front montant de l’entrée 7.
```

C) Instruction: TRIGGERC

Cette instruction indique que le prochain mouvement sera déclenché sur un numéro de capture.

Syntaxe : TRIGGERC (<N° de capture>)

< N° de capture > de 1 à 2.

Exemple :

```
STTA =50 ‘mouvement absolu en 50 sans attente de fin de mouvement
…
CAPTURE1(0,4,On,10,20,On)     'Capture position sur front montant
                              ‘de l’entrée 4 lorsque l’axe du
                              ‘moteur est entre 10 et 20.
TRIGGERC (1)
STTA =300     ‘mouvement absolu en 300
```

‘ déclenché sur déclenchement de la capture 1.

Attention : l’exécution de l’instruction TRIGGERC, annule la capture, il est donc possible de recharger une nouvelle capture

Le TRIGGERC avec les entrées 3, 4, 15, 16 (entrées rapides) fonctionnent comme avec des entrées standards.

D) Instruction: TRIGGERS

Cette instruction déclenche le mouvement déclenché sans aucune condition.

Cette instruction doit être utilisée dans une tâche parallèle à celle contenant l’instruction TRIGGER.

E) Instruction: TRIGGERR

Cette instruction annule le mouvement déclenché sans aucune condition.

Cette instruction doit être utilisée dans une tâche parallèle à celle contenant l’instruction TRIGGER.

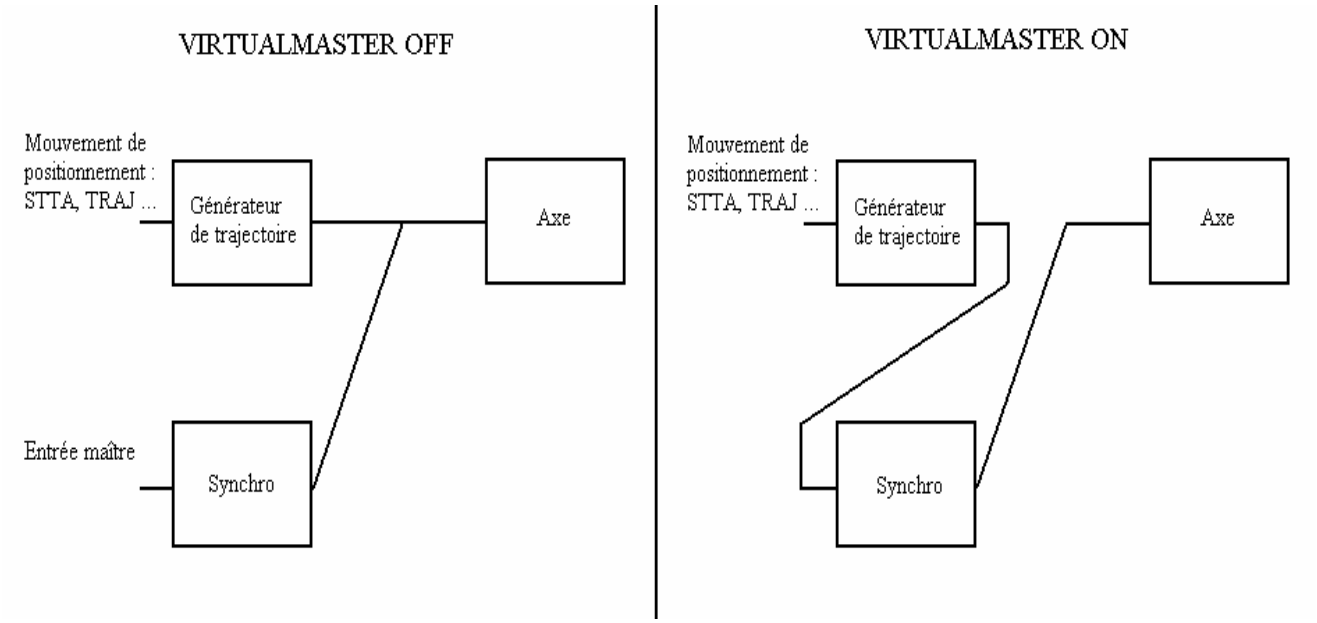
7-10- Maître virtuel

7-10-1- Maître virtuel

Il est possible de passer l’axe maître en mode virtuel afin de développer une application sans avoir de maître.

VIRTUALMASTER – Active/désactive le maître virtuel

Syntaxe: VIRTUALMASTER ON/OFF



Description : Cette instruction permet de déclarer l'axe maître en mode virtuel. Toutes les instructions de positionnement MOVA, MOVR, STTA, SSTR seront « exécutées par le maître », l'axe maître se « déplacera virtuellement ». Il est possible de réaliser des fonctions de synchronisations entre le maître et l'esclave (le moteur) en utilisant les instructions MOVS, GEARBOX ….

Attention : Pour utiliser cette instruction, sélectionner « virtuel » comme source du maître dans la fenêtre **Motion control \ Maître/esclave.**

L'unité utilisée est celle de l'entrée maître.

MOVEMASTER_S – Indique si un mouvement est en cours lorsqu'on est en maître virtuel

Syntaxe : MOVEMASTER_S

Types acceptés : Bit

Description : MOVEMASTER_S est égal à 0 si les 3 points suivants sont vrais :

1. On est en maître virtuel.
2. Le mouvement de positionnement (STTA, TRAJ…) courant est terminé (trajectoire théorique terminée).
3. Le buffer de mouvement de positionnement (STTA, TRAJ…) est vide.

Exemple :

```
VIRTUALMASTER ON
STTA = VR10
WAIT MOVEMASTER_S = OFF     ‘ Attente que la trajectoire sur
                            ‘ le maître virtuel soit arrêté
```

STOPMASTER – Arrête le mouvement du maître virtuel

Syntaxe : STOPMASTER

Description : Cette fonction arrête le mouvement du maître virtuel. La fonction est bloquante tant que le mouvement n'est pas arrêté.

Remarques : Si l'axe est un axe lié avec un mouvement de synchronisation, alors l'axe s'arrête.

L'instruction STOPMASTER vide le buffer de mouvement du maître et stoppe l'axe en utilisant la décélération courante. Cette instruction est bloquante tant que MOVEMASTER_S est différent de 0.

Exemple :

```
VIRTUALMASTER ON
MOVS (1, 1, 0, 0)
STTA = 10
…                    ‘ le maître s’arrête, l’axe ne tourne plus
STOPMASTER           ‘ mais le mouvement synchro est toujours actif
STTA = 10            ‘ le maître démarre et l’axe recommence à tourner
```

SSTOPMASTER – Arrête le mouvement du maître virtuel sans attente

Syntaxe : SSTOPMASTER

Description : Cette fonction arrête le mouvement du maître virtuel. La fonction n'est pas bloquante pour la tâche.

Remarques : Si l'axe est lié avec un mouvement de synchronisation alors l'axe s'arrête.

L'instruction SSTOPMASTER vide le buffer de mouvement du maître et stoppe l'axe en utilisant la décélération courante. Cette instruction n'est pas bloquante et n'attend pas que MOVEMASTER_S soit égal à 0.

Exemple :

```
VIRTUALMASTER ON
MOVS (1, 1, 0, 0)
STTA = 10
…
SSTOPMASTER          ‘ Demande d’arrêt du maître
WAIT MOVEMASTER_S = 0 ‘ Attente fin de mouvement du maître
STTA = 10            ‘ le maître redémarre et l’axe recommence à tourner
```

8- Programmation de l'automate

8-1- Entrées/Sorties logiques

8-1-1- Lecture des entrées

La fonction INP est utilisée pour lire 1 bit, INPB un bloc de 8 bits et INPW un bloc de 16 bits.

Les syntaxes sont : INP (<NuméroEntrée>), INPB (<NuméroBloc>), INPW

<NuméroEntrée> doit représenter le numéro d'une entrée et <NuméroBloc> le numéro d'un bloc de 8 entrées. Ce numéro correspond au numéro de l'entrée dans le module de configuration. Le type de données retourné est :

- Bit pour une entrée
- Octet pour un bloc de 8 entrées
- Entier pour un bloc de 16 entrées

Par exemple :

```
VF1= INP (3) 'lecture d'une entrée n°3
VB2 = INPB (1)        'lecture du premier bloc de 8 entrées
VB4 = INPB (2)        'lecture du deuxième bloc de 8 entrées
VI3= INPW    'lecture des 16 entrées
```

8-1-2- Ecriture des sorties

La fonction OUT est utilisée pour écrire 1 bit, OUTB un bloc de 8 bits.

Les syntaxes sont : OUT (<NuméroSortie>), OUTB (<NuméroBloc>).

<NuméroSortie>doit représenter le numéro d'une sortie ou <NuméroBloc> le numéro d'un bloc de 8 sorties. Ce numéro correspond au numéro dans le module de configuration. Le type de données utilisé est :

- Bit pour une sortie
- Octet pour un bloc de 8 sorties

Par exemple :

```
OUT (5) = 1          'Mise à 1 de la sortie n°5
OUTB (1) = 48        'écriture d'un bloc de 8 sorties
```

8-1-3- Lecture des sorties

Toutes les sorties peuvent également être lues. La valeur lue est la dernière valeur écrite. Cette caractéristique est très utile quand plus d'un programme utilise le même bloc de sorties. Donc, il est possible d'écrire seulement les sorties désirées dans une opération sans changer les autres.

Par exemple :

Pour mettre à 1 le quatrième bit d'un bloc de 8 bits :

```
OUTB (2)= 16         'mise à 1 du quatrième bit du bloc n°2 de 8 bits
VB0 = OUTB (2)
```

8-1-4- Attente d'un état

Il est possible d'attendre un changement d'état sur une entrée grâce à l'instruction WAIT.

La syntaxe est : WAIT <Condition>

La fonction WAIT est utilisée pour attendre une condition de changement durant une exécution normale. L'exécution est stoppée aussi longtemps que la condition est fausse. Quand l'état devient vrai, l'exécution continue. Cette fonction est très utile pour attendre la fin des mouvements ou une butée logicielle...

Exemple :

```
WAIT INP (2) = ON        'Attente que l'entrée 2 soit à 1
STOP                     'Arrêt de l'axe
WAIT INP (5) = ON        'Attente que entrée n°5 soit à 1
```

8-1-5- Test d'un état

Il est possible de tester l'état d'une entrée grâce à l'instruction IF…

La syntaxe est : IF (<Condition>) GOTO <Etiquette>

La structure IF… est utilisée pour tester une condition à un instant donné. La validation de la <Condition> permet de réaliser un branchement à une étiquette.

Exemple :

```
IF INP (5) = ON GOTO Suite_OK        'Test de l'état de l'entrée n°5,
                                     'si entrée à 1 saut en Suite_OK
```

8-2- Entrées/Sorties analogiques

8-2-1- Lecture d'une entrée

Les fonctions ADC (1) et ADC (2) sont utilisées pour lire 2 entrées analogiques. Les données retournées par la fonction sont toujours de type réel et comprises entre -10 et +10.

Par exemple:

```
VR1 = ADC(1)        'Lecture de l'entrée analogique 1
VR5 = ADC(2)        'Lecture de l'entrée analogique 2
```

8-2-2- Ecriture d'une sortie

La fonction DAC est utilisée pour écrire sur la sortie analogique.

La syntaxe est : DAC=<Expression réelle>

Les données utilisées par l'instruction sont toujours de type réel et comprises entre -10 et +10.

Par exemple :

```
DAC=5.0                    'Ecriture d'une valeur de consigne de 5 V
```

8-3- Temporisations

8-3-1- Attente passive

La fonction DELAY est utilisée pour établir une attente passive. Sa syntaxe est :

DELAY <Durée>

<Durée> est un entier exprimé en milliseconde. Il est recommandé d'utiliser cette fonction pour une longue attente passive car le programme en attente ne prend pas de temps processeur.

Avec cette fonction, le programme attend la durée indiquée.

Par exemple:

```
Debut:
    WAIT INP(5) = 1
    ...
    DELAY 5000              ' Délai de 5 secondes
    ...
GOTO Debut
```

Remarque : L'utilisation des instructions SAVEPARAM et SAVEVARIABLE fausse la base de temps.

8-3-2- Attente active

a) TIME :

La variable globale interne TIME peut être utilisée pour établir des attentes actives. TIME est un entier long qui représente le millième de secondes écoulées depuis la dernière mise sous tension. Cette variable peut donc être utilisée comme base de temps. Elle convient en particulier aux machines qui sont sous-tension moins de 25 jours. En effet à la mise sous-tension, TIME est initialisé à 0. Au-delà de 25jours, la variable atteint sa valeur maximum 2^31 et passe ensuite à 2^-31. Cette transition appelée débordement peut provoquer dans

certain cas des erreurs de temporisations, pour éviter ce problème il est préférable d'utiliser l'instruction LOADTIMER.

Par exemple :

```
VL2=TIME

VL2=VL2+5000

ATTENTE:

  VL3=TIME

  IF VL3<VL2 GOTO ATTENTE              'Temporisation de 5s
```

Remarque : TIME est de type entier long

Attention : La fonction TIME ne fonctionne pas dans un test

b) LOADTIMER et TIMER :

L'instruction LOADTIMER peut être utilisée pour établir des attentes actives. C'est un réel qui représente le millième de secondes écoulées depuis la dernière mise sous tension. Cette variable peut donc être utilisée comme base de temps. Elle convient en particulier aux machines qui sont toujours sous-tension.

Elle permet également de charger dans un timer une valeur, qui se décrémentera automatiquement jusqu'à 0. Il est possible de savoir si le timer est écoulé en utilisant l'instruction TIMER (VLXX), avec XX compris entre 0 et 255.

Si TIMER (VLXX) = 1 la temporisation n'est pas écoulée.

Si TIMER (VLXX) = 0 la temporisation est écoulée.

Il est possible d'utiliser simultanément 256 timers.

Par exemple :

```
LOADTIMER (VL129)=3000               'Chargement d'une temporisation de 3s

BOUCLE :

        IF TIMER (VL129) <>0 GOTO BOUCLE          'Attente de la fin de la tempo
```

Remarque : Pendant l'exécution de ces lignes la variable VL129 de type entier long est utilisée par le système

L'utilisation des instructions SAVEPARAM et SAVEVARIABLE fausse la base de temps.

8-4- Compteurs

8-4-1- Compteurs

Attention :

- Il est interdit d'utiliser simultanément la même entrée et le même front sur les fonctions de mouvement déclenché, les captures et les compteurs.

- Lorsque le compteur atteint sa valeur maxi, il repasse à 0 au prochain front (valeur maxi : 65535).

A) Configuration :

L'instruction SETUPCOUNTER permet de configurer le compteur.

Syntaxe : SETUPCOUNTER (<n° de compteur>, <Entrée>, <Filtre>)

<n° de compteur> : 0 ou 1

<Entrée> : Numéro de l'entrée (1 à 16)

<Filtre> : Validation du filtre : 0 pour sans filtrage, 1 pour avec filtrage.

Si le filtre n'est pas activé, la fréquence maxi est de 5 KHz sinon il dépend du paramètre Filtrage dans **Paramètres** / **Entrées Sorties Digitales**.

B) Ecriture :

L'instruction COUNTER(1 ou 2) permet d'initialiser le compteur à une valeur.

Syntaxe : COUNTER(<n° de compteur>) = <Val>

<n° de compteur> : Numéro de compteur (1 ou 2)

<Val> : Valeur comprise entre 0 et 65535

C) Lecture :

L'instruction COUNTER_S permet de lire le compteur.

Syntaxe : <Variable>=COUNTER_S(<n° de compteur>)

<Variable> : entier compris entre 0 et 65535

<n° de compteur>: Numéro de compteur (1 ou 2)

8-5- Boîte à cames

8-5-1- Boîte à cames

Les boîtes à cames permettent de piloter des sorties logiques suivant des positions angulaires, linéaires par des instructions optimisées.

iDPL accepte 2 boîtes à cames avec jusqu'à 4 segments par boîte. Par exemple, les sorties 3, 4 et 12 peuvent être affectées à la boîte et les autres sorties peuvent être utilisées ailleurs.

Les sorties d'une boîte sont remises à jour toutes les 300µs.

Les fonctions disponibles sont :

CAMBOX, CAMBOXSEG, STARTCAMBOX et STOPCAMBOX

Lors de la déclaration d'un segment, la valeur de début peut-être supérieure à celle de fin. Le zéro programme est pris en compte à chaque définition de segment.

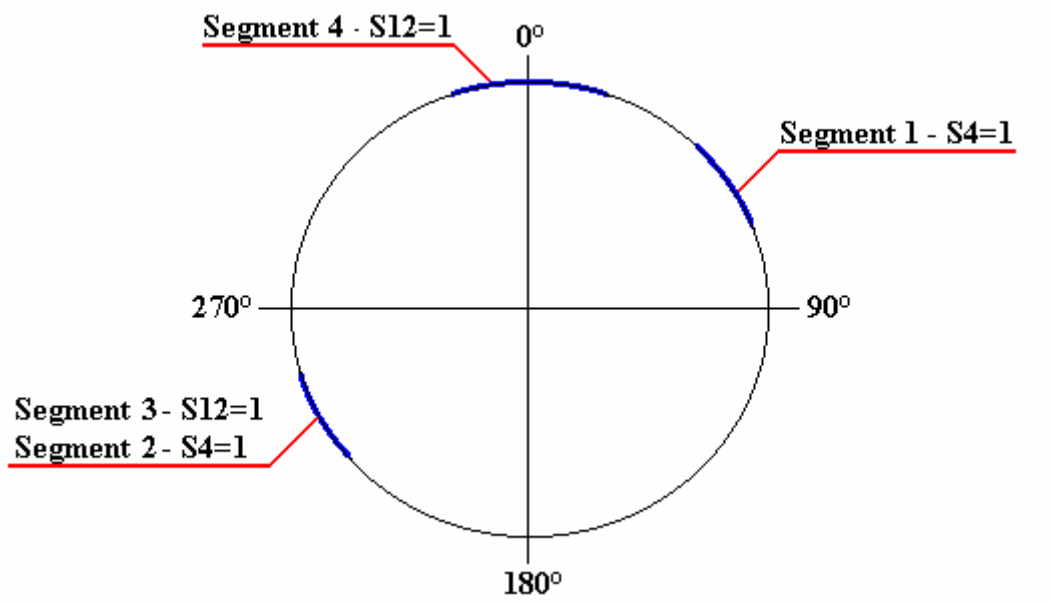
Le variateur gère jusqu'à deux boites à cames de quatre segments chacune.

La source est soit la position du codeur moteur, soit la position du codeur maître (connecteur X2).

Dans le cas où la source est le moteur, les valeurs de début et de fin de segment sont directement liées à l'unité et à la mise à l'échelle paramétrées dans l'écran **Motion control** / **Configuration** / **Unités.**

Dans le cas où la source est le codeur maître, les valeurs de début et de fin de segment sont directement liées à l'unité et à la mise à l'échelle paramétrées dans l'écran **Motion control** / **Configuration** / **Maître.**

Dans l'instruction CAMBOXSEG, les début et fin de segments devront être compris entre 0° et la valeur du modulo.



Dans cet exemple, le codeur maître est modulo 360. La boîte à cames s'écrit de la façon suivante :

CAMBOX (1,1,4) codeur maître	'La boîte à cames n°1 à 4 segments, source
CAMBOXSEG(1,1,4,40,60) entre 40° et 60°	'Le segment 1 de la boîte n°1 met la sortie 4 à 1
CAMBOXSEG(1,2,4,230,250) entre 230° et 250°	'Le segment 2 de la boîte n°1 met la sortie 4 à 1
CAMBOXSEG(1,3,12,230,250) entre 200° et 400°	'Le segment 3 de la boîte n°1 met la sortie 12 à 1
CAMBOXSEG(1,4,12,350,10) entre 350° et 10°	'Le segment 4 de la boîte n°1 met la sortie 12 à 1
STATCAMBOX(1)	‘Démarrage de la boîte n°1
…	
STOPCAMBOX (1)	'Arrêt de la boîte n°1

9- Liste des opérateurs et instructions

Pour connaître le temps d'exécution de chaque instruction, consulter le fichier iDPL TIME INSTRUCTION.XLS dans le répertoire \Data du CD.

9-1- Programme

CALL	Appel de Sous-programme
NEXTTASK	Basculement immédiat à la tâche suivante
GOTO	Saut à une étiquette
PROG ... END PROG	Début d'un programme
SUB ... END SUB	Sous-programme
EXIT SUB	Sortie d'un sous-programme

9-2- Arithmétique

+	Addition
-	Soustraction
*	Multiplication
/	Division

9-3- Mathématique

ARCCOS	Cosinus inverse
ARCSIN	Sinus inverse
ARCTAN	Tangente inverse
COS	Cosinus
EXP	Exponentiel
FRAC	Partie fractionnelle
LOG	Logarithme
INT	Partie entière

MOD	Modulo
SGN	Signe
SIN	Sinus
SQR	Racine carrée
TAN	Tangente

9-4- Logique

<<	Décalage à gauche
>>	Décalage à droite
AND	Opérateur ET
NOT	Opérateur complément
OR	Opérateur OU
XOR	Opérateur OU exclusif

9-5- Test

<	Inférieur
<=	Inférieur ou égal
<>	Différent
=	Egal / affectation
>	Supérieur
>=	Supérieur ou égal
IF	Test conditionnel

9-6- Contrôle de mouvement

A) Contrôle de l'axe :

ACC	Accélération
ACC%	Accélération en pourcentage
AXIS	Contrôle la boucle d'asservissement
AXIS_S	Lit l'état de la boucle d'asservissement
BUFMOV_S	Nombre d'ordres en attente
CLEAR	Met à zéro la position de l'axe
CLEARMASTER	Met à zéro la position de l'axe maître
DEC	Décélération
DEC%	Décélération en pourcentage
FE_S	Erreur de poursuite
FEMAX_S	Limite d'erreur de poursuite
HOME	Prise d'origine
HOME_S	Etat de la prise d'origine
HOMEMASTER	Prise d'origine sur le maître
HOMEMASTER_S	Etat de la prise d'origine sur le maître
LOOP	Mode virtuel
MERGE	Définit l'enchaînement
MOVE_S	Etat du mouvement
ORDER	Numéro d'ordre du mouvement
ORDER_S	Numéro d'ordre courant
POS	Position à atteindre
POS_S	Position réelle
POSMASTER_S	Position réelle du maître
VEL	Vitesse
VEL_S	Retourne la vitesse courante
VEL%	Vitesse en pourcentage
VELMASTER_S	Retourne la vitesse courant du maître

B) Positionnement :

MOVA	Mouvement absolu
MOVR	Mouvement relatif
SSTOP	Arrêt d'un axe sans attente
STOP	Arrêt d'un axe
STTA	Lance un mouvement absolu
STTI	Lance un mouvement infini
STTR	Lance un mouvement relatif

C) Synchronisation :

CAMNUM_S	Numéro de la came en cours d'exécution
CAMREADPOINT	Position de l'esclave dans la came
CAMSEG_S	Numéro d'équation de la came en cours d'exécution
ICORRECTION	Fonction de compensation
ENDCAM	Arrêt d'une came
FILTERMASTER	Applique un filtrage lors de mouvement synchrone
ICORRECTION_S	Etat de la compensation
GEARBOX	Arbre électrique
GEARBOXRATIO	Modifie le rapport de réduction d'un arbre électrique
LOADCAM	Charge une came dans la variateur
LOADCAMPOINT	Modification de points d'une came
MASTEROFFSET	Décale dynamiquement la position du maître
MOVS	Permet d'effectuer une synchronisationn
READCAM	Permet de lire un point de came
SLAVEOFFSET	Décale dynamiquement la position de l'esclave
STARTCAM	Exécute une came
STARTGEARBOX	Lance l'arbre électrique

STOPS	Permet de stopper une synchronisation en cours
STOPS_S	Etat du mouvement synchronisé
WRITECAM	Permet d'écrire un point de came

D) Capture

CAPTURE1 et CAPTURE2	Lancement de capture de position
DISABLERECALE	Désactivation du recalage
ENABLERECALE	Fonction de recalage automatique
REGPOS1_S et REGPOS2_S	Position capturée
REG1_S et REG2_S	Etat de la capture

9-6-2- Mouvements déclenchés

TRIGGERP	Triggerise le prochain mouvement sur le passage du maître à une position.
TRIGGERI	Triggerise le prochain mouvement sur changement d'état d'une entrée.
TRIGGERC	Ttriggerise le prochain mouvement sur un numéro de capture.
TRIGGERS	Déclenche le mouvement triggé sans aucune condition.
TRIGGERR	Annule le mouvement triggé sans aucune condition.

9-6-3- Maître virtuel

MOVEMASTER_S	Active/désactive le maître virtuel
SSTOPMASTER	Arrête le mouvement du maître virtuel sans attente
STOPMASTER	Arrête le mouvement du maître virtuel
VIRTUALMASTER	Active/désactive le maître virtuel

9-7- Automate

A) Entrées / sorties TOR

CAMBOX	Boîte à cames
CAMBOXSEG	Segment de boîte à cames
INP	Lecture d'une entrée logique
INPB	Lecture d'un bloc de 8 entrées
INPW	Lecture d'un bloc de 16 entrées
OUT	Ecriture d'une sortie
OUTB	Ecriture d'un bloc de 8 sorties
STARTCAMBOX	Lance une boîte à cames
STOPCAMBOX	Arrête une boîte à cames
WAIT	Attente d'une condition

B) Entrées / sorties analogiques

ADC(1)	Entrée analogique n°1
ADC(2)	Entrée analogique n°2
DAC	Sortie analogique

C) Temporisations

DELAY	Attente passive
LOADTIMER	Charge une temporisation dans une variable
TIME	Base de temps
TIMER	Compare une variable à Time

D) Compteurs

COUNTER	Initialise un compteur à une valeur
SETUPCOUNTER	Configuration du compteur
COUNTER_S	Renvoie la valeur d'un compteur

9-8- Gestion des tâches

CONTINUE	Continue l'exécution d'une tâche
HALT	Arrête une tâche
RUN	Lance une tâche
SUSPEND	Suspend une tâche
STATUS	Etat d'une tâche

9-9- Flash, Sécurité, Divers

DISPLAY	Afficheur 7 segments
LOADPARAM en FLASH	Permet de recharger les paramètres du variateur
LOADVARIABLE FLASH	Permet de charger les variables sauvegardées en
READI	Permet de lire un entier en FRAM
READL	Permet de lire un entier long en FRAM
READR	Permet de lire un réel en FRAM
RESTART	Redémarrage du variateur
SAVEPARAM	Permet de sauvegarder les paramètres du variateur en FLASH
SAVEVARIABLE	Permet de sauvegarder les variables VR0..VR63, VL0..VL63 en FLASH
SECURITY	Définit les actions de sécurité
VERSION	Renvoie la version de l'Operating System
WRITEI	Permet d'écrire un entier en FRAM
WRITEL	Permet d'écrire un entier long en FRAM
WRITER	Permet d'écrire un réel en FRAM
COMCOUNTER	Retourne le nombre de trames échangées

9-10- Liste aplhabétique

9-10-1- Addition (+)

Syntaxe : <Expression1> + <Expression2>

Types acceptés :Octet, Entier, Entier long et réel

Description : Cet opérateur additionne deux expressions et retourne une valeur du même type que ces opérandes.

Remarques : <Expression1> et <Expression2> doivent être des expressions valides. <Expression1> et <Expression2> doivent être de même type.

Exemple : VL1=10

VL2=5

VL3=VL1+VL2 'Résultat : VL3=15

Voir aussi : `-`, `*' et `/'.

9-10-2- Soustraction (-)

Syntaxe : <Expression1> - <Expression2>

Types acceptés :Octet, Entier, Entier long ou réel

Description : Cet opérateur soustrait l'<Expression2> de l'<Expression1> et retourne une valeur du même type que ces opérandes.

Remarques : <Expression1> et <Expression2> doivent être des expressions numériques valides. <Expression1> et <Expression2> doivent être de même type.

Exemple : VL1=10

VL2=5

VL3=VL1-VL2 'Résultat : VL3=5

Voir aussi : '+', `*' et `/'.

9-10-3- Multiplication (*)

Syntaxe : <Expression1> * <Expression2>

Types acceptés :Octet, Entier, Entier long ou réel

Description : Cet opérateur multiplie l'<Expression1> par l'<Expression2> et retourne une valeur du même type que ces opérandes.

Remarques : <Expression1> et <Expression2> doivent être des expressions numériques valides. <Expression1> et <Expression2> doivent être de même type.

Exemple : VL1=10

VL2=5

VL3=VL1*VL2 'Résultat : VL=50

Voir aussi : '+', '-' et `/'.

9-10-4- Division (/)

Syntaxe : <Expression1> / <Expression2>

Types acceptés :Octet, Entier, Entier long ou réel

Description : Cet opérateur divise l'<Expression1> par l'<Expression2>

Remarques : <Expression1> et <Expression2> doivent être des expressions numériques valides. <Expression1> et <Expression2> doivent être de même type. <Expression2> doit être différente de zéro. Cet opérateur retourne toujours une valeur réelle.

Exemple : VL1=10

VL2=5

VL3=VL1/VL2 'Résultat : VL3=2

Voir aussi : '+', '-', `*'.

9-10-5- Inférieur (<)

Syntaxe : <Expression1> < <Expression2>

Types acceptés :Bit, Octet, Entier, Entier long ou réel

Description : Cet opérateur teste si <Expression1> est inférieure à <Expression2>.

Remarques : <Expression1> et <Expression2> doivent être des expressions valides. <Expression1> et <Expression2> doivent être de même type.

Exemple : VL1=10

IF VL1 < VL 2 ...

Voir aussi : '=', '>', '>=', '<=', '<>'.

9-10-6- Inférieur ou égal (<=)

Syntaxe : <Expression1> <= <Expression2>

Types acceptés :Bit, Octet, Entier, Entier long, réel

Description : Cet opérateur teste si <Expression1>est inférieure ou égale à <Expression2>.

Remarques : <Expression1> et <Expression2> doivent être des expressions valides. <Expression1> et <Expression2> doivent être de même type.

Exemple : VL1 =10

```
IF VL1<= VL1 ...
```

Voir aussi : '=', '>', '>=', '<', '<>'.

9-10-7- Décalage à gauche (<<)

Syntaxe : <Expression1> << <Expression2>

Types acceptés: Octet ou Entier

Description : Cet opérateur déplace <Expression2> bits de <Expression1> de droite à gauche.

Remarques : <Expression2> représente le nombre de bits à déplacer. Le décalage n'est pas circulaire.

Exemple : VL1 = 4

```
VL2= VL1 << 2       'Résultat VL2= 16
```

Voir aussi : ' >> '.

Attention : Laisser un espace avant et après le décalage.

9-10-8- Différent (<>)

Syntaxe : <Expression1> <> <Expression2>

Types acceptés :Bit, Octet, Entier, Entier long, réel

Description : Cet opérateur teste si <Expression1> et <Expression2> sont différentes.

Remarques : <Expression1>et <Expression2> doivent être des expressions valides. <Expression1> et <Expression2> doivent être de même type.

Exemple : VL1=10

IF VL2<> VL1 ...

Voir aussi : '=', '>', '>=', '<', '<='

9-10-9- Affectation/Egalité (=)

Syntaxe : <Expression1> = <Expression2> Ou <Variable>=<Expression2>

Types acceptés :Bit, Octet, Entier, Entier long, réel

Description : Cet opérateur affecte <Variable> à <Expression2> ou teste si <Expression1> est égale à <Expression2>.

Remarques : <Expression1> et <Expression2> doivent être des expressions valides. <Expression1>, <Expression2> et <Variable> doivent être de même type.

Exemple : VL1=1

BOUCLE :

VL1 = VL1 + 1

IF VL1 =10 GOTO SUITE

GOTO BOUCLE

SUITE :

Voir aussi : '>', '>=', '<', '<=', '<>'

9-10-10- Supérieur (>)

Syntaxe : <Expression1> > <Expression2>

Types acceptés :Bit, Octet, Entier, Entier long, réel

Description : Cet opérateur teste si <Expression1> est supérieure à <Expression2>.

Remarques : <Expression1> et <Expression2> doivent être de même type.

Exemple : IF VL1 > VL2 ...

Voir aussi : '=', '>=', '<', '<=', '<>'

9-10-11- Supérieur ou égal (>=)

Syntaxe : <Expression1> >= <Expression2>

Types acceptés :Bit, Octet, Entier, Entier long, réel

Description : Cet opérateur teste si <Expression1> est supérieure ou égale à <Expression2>.

Remarques : <Expression1> et <Expression2> doivent être de même type.

Exemple : IF VL1 >= VL2 ...

Voir aussi : '=', '>', '<', '<=', '<>'.

9-10-12- Décalage à droite (>>)

Syntaxe : <Expression1> >> <Expression2>

Types acceptés :Octet ou Entier

Description : Cet opérateur déplace <Expression2> bits de <Expression1> de gauche à droite.

Remarques : <Expression2> représente le nombre de bits à déplacer. Le décalage n'est pas circulaire.

Exemple : VL1 = 48

VL2 = VL1 >> 3 'Résultat VL2 = 12

Voir aussi : ' << '.

Attention : Laisser un espace avant et après le décalage.

9-10-13- ACC - Accélération

Syntaxe 1 : ACC = <Expression>

Syntaxe 2 : <Variable> = ACC

Unité : Expression,Variable : unité utilisateur par s² (Ex : mm/s², degré/s², tr/s²…)

Types acceptés :<Expression> : réel

<Variable> : réel

Description : Cette instruction lit ou modifie l'accélération courante.

Remarques : <Expression> doit être une expression réelle valide. L'accélération courante peut être lue et modifiée à tout moment.

Exemple : ACC = 500

VR0 = 1000

ACC = VR0

Voir aussi : DEC, POS et VEL

9-10-14- ADC(1) – Entrée analogique 1

Syntaxe : <Variable>= ADC(1)

Unité : Variable : Volt

Limite : Variable : +/- 10V

Types acceptés :<Variable> : réel

Description : Cette fonction retourne la tension de l'entrée analogique n°1.

Exemple : VR1=ADC(1)

Voir aussi : DAC, ADC(2)

9-10-15- ADC(2) – Entrée analogique 2

Syntaxe : <Variable>= ADC(2)

Unité : Variable : Volt

Limite : Variable : +/- 10V

Types acceptés :<Variable> : réel

Description : Cette fonction retourne la tension de l'entrée analogique n°2.

Exemple : VR2 =ADC(2)

Voir aussi : DAC, ADC(1)

9-10-16- ACC% - Accélération en pourcentage

Syntaxe : ACC% = <Expression>

Limites : Expression : de 1 à 100

Types acceptés :<Expression> : Octet

Description : Cette fonction ajuste l'accélération courante en pourcentage du paramètre d'accélération.

Remarques : La valeur du paramètre accélération peut être entrée dans l'écran Motion control / Configuration / Profil de vitesse.

Exemple : ACC%=10 ' L'accélération courante est de 10%

VB = 50

ACC%=VB0

Voir aussi : DEC%

9-10-17- AND – Opérateur ET

Syntaxe : <Expression1> AND <Expression2>

Types acceptés :Bit, Octet ou entier

Description : Cette fonction effectue un ET binaire entre deux expressions et retourne une valeur du type de l'opérande.

Remarques : <Expression1> et <Expression2> doivent être du même type.

Exemple : VB3=1001111b

VB4=1111110b

VB2=VB3 AND VB4 'VB2=1001110b

Voir aussi : OR, NOT, XOR et IF

9-10-18- ARCCOS – Cosinus inverse

Syntaxe : ARCCOS (<Expression>)

Limite : de –1 à +1

Types acceptés :Octet, Entier, Entier long, réel

Description : Cette fonction restitue l' arccosinus de <Expression>.

Remarques : Cette fonction restitue un angle exprimé en radian.

Exemple : VR1=ARCCOS(VR0)

Voir aussi : SIN, COS et TAN

9-10-19- ARCSIN – Sinus inverse

Syntaxe : ARCSIN (<Expression>)

Limite : de –1 à +1

Types acceptés :Octet, Entier, Entier long, réel

Description : Cette fonction restitue l' arcsinus de <Expression>.

Remarques : Cette fonction restitue un angle exprimé en radian.

Exemple : VR1=ARCSIN(1)

Voir aussi : SIN, COS et TAN

9-10-20- ARCTAN – Tangente inverse

Syntaxe : ARCTAN (<Expression>)

Types acceptés :Octet, Entier, Entier long, réel

Description : Cette fonction restitue l'arctangente de <Expression>.

Remarques : La fonction ARCTAN prend le rapport de deux côtés d'un triangle rectangle et restitue l'angle correspondant. Le rapport est la longueur du côté opposé à l'angle par la longueur du côté adjacent à l'angle.

Exemple : VR0=ARCTAN(3)

VR1=ARCTAN(1)

Voir aussi : SIN, COS et TAN

9-10-21- AXIS – Contrôle la boucle d'asservissement

Syntaxe : AXIS ON | OFF

Description : Cette instruction est utilisée pour ouvrir et fermer la boucle d'asservissement.

Remarques : Lorsque l'axe est en boucle fermée (AXIS ON), toutes les instructions de mouvement sont transmises à l'axe par l'intermédiaire du buffer de mouvement et sont exécutées. Si l'axe passe en boucle ouverte (AXIS OFF), le buffer de mouvement est vidé, les instructions MOVE_S et FE_S retournent la valeur 0.

Exemple :
```
AXIS ON              'passage en boucle fermée
    MOVA=1000        'déplacement à la position 1000
    OUT(3)=1         'Sortie n°3 =1
    MOVA=2000
    OUT(3)=0
```

Attention : Voir le mode de déverrouillage variateur dans l'écran Paramètres / Entrées Sorties Digitales.

Voir aussi : AXIS_S, SECURITY

9-10-22- AXIS_S – Lit l'état de la boucle d'asservissement

Syntaxe : AXIS_S

Description : Cette fonction est utilisée pour lire l'état de la boucle d'asservissement et retourne une réponse 1 ou 0.

Remarques : Cette fonction peut être utilisée à tout moment pour voir si l'axe est asservi.

Exemple : MOVA=100

If AXIS_S = 0 GOTO Error 'Erreur car l'axe est passé en 'boucle ouverte

Voir aussi : AXIS

9-10-23- BUFMOV_S

Syntaxe : <Variable>=BUFMOV_S

Types acceptés : <Variable> : Octet

Description : Cette fonction retourne le nombre de mouvements en attente dans le buffer du variateur. Le mouvement en cours d'exécution n'est pas comptabilisé par cette fonction.

Remarques : Cette fonction peut être utilisée après avoir lancé des mouvements, pour savoir si un mouvement est fini. Dans le cas où le buffer de mouvement se trouve plein, la tâche se bloque jusqu'à ce qu'une place soit libérée.

Exemple : STTR=100

STTR=50

STTR=50

WAIT BUFMOV_S<2 'Attendre la fin du premier mouvement

9-10-24- CALL – Appel d'un sous-programme

Syntaxe : CALL <Nom>

Description : Cette instruction est utilisée pour appeler un sous-programme défini par un bloc SUB. <Nom> est le nom du bloc du sous-programme.

Remarques : Un sous-programme ne peut pas s'appeler. L'exécution de cette instruction provoque un basculement de tâche.

Exemple : CALL Mouvement

Voir aussi : SUB

9-10-25- CAMBOX – Boîte à cames

Syntaxe : CAMBOX (<Num boîte>, <Source>, <Nb Seg>)

Limites : Num boîte : de 1 à 2

Source : 0 si moteur ou 1 si codeur maître.

Nb seg : de 1 à 4

Types acceptés :Num boîte : Octet

Nb Seg : Octet

Description : Cette fonction définit une boîte à cames. Tout segment (CAMSEG) précédemment défini sur cette boîte est effacé.

Remarques : <Num boite> désigne un numéro de boîte

<Nb seg> est le nombre de segment dans la boîte. Si cette valeur est nulle la boîte à came est détruite et doit être redéfinie si on veut la réutiliser.

Exemple : CAMBOX(1,1,4) 'Boîte à came n°1 de 4 segments dont la source est un codeur maître

Voir aussi : CAMBOXSEG

9-10-26- CAMBOXSEG – Segment de boîte à cames

Syntaxe : CAMBOXSEG (<Num boîte>, < Num Seg >, <Num sortie>, <Début>,<Fin>)

Limites : Num boîte : de 1 à 2

Num Seg : de 1 à 4

Num sortie : de 1 à 10

Unité : Début, Fin : Unité utilisateur moteur

Types acceptés :Num boîte, Num seg, Num sortie : Octet

Début, Fin :réel

Description : Cette fonction définit un segment d'une boîte à came.

Remarques : <Num boîte> désigne la boîte. <Num seg> désigne le segment. <Num sortie> est la sortie à modifier. La sortie sera mise à 1 entre <Début> et <Fin>.

Exemple : CAMBOXSEG(1,2,4,0,90) 'Le second segment de la boîte 1 met la 4ème sortie à 1 entre 0 et 90°(l'unité utilisateur définie étant le degré).

Voir aussi : CAMBOX

9-10-27- CAMNUM_S – Numéro de la came en cours d'exécution

Syntaxe : <Variable> = CAMNUM_S

Types acceptés :<Variable> : Entier

Description : Cette instruction permet de savoir quel numéro de came est en cours d'exécution.

Remarques : La valeur retournée est significative que si CAM_S est à 1.

Exemple :
```
IF CAMNUM_S=1 THEN GOTO ATTENTE_FIN_CAME_1
      'Came 1 en cours

IF CAMNUM_S=2 THEN GOTO ATTENTE_FIN_CAME_2
      'Came 2 en cours
```

Voir aussi : CAM_S, CAMSEG_S

9-10-28- CAMREADPOINT – Position de l'esclave dans la came

Syntaxe : <Position Esclave> = CAMREADPOINT (<Position Maître>, <N° came>)

Description : Cette instruction permet de calculer la position de l'esclave <Position Esclave> dans la came, correspondant à la position du maître <Position Maître>.

Types acceptés : <Position Esclave> type réel dans l'unité de l'esclave

<Position Maître> type réel dans l'unité du maître

<N° came> : Numéro de la came cible chargée précédemment (de 1 à 5).

Remarque : Retourne 0 si on n'est pas dans la came sélectionnée.

9-10-29- CAMSEG_S – Numéro d'équation de la came en cours d'exécution

Syntaxe : <Variable> = CAMSEG_S

Types acceptés :<Variable> : Entier

Description : Cette instruction permet de savoir quel numéro d'équation de la came est en cours d'exécution.

Remarques : La valeur retournée est significative que si CAM_S est à 1.

Exemple : IF CAMSEG_S=1 THEN GOTO ATTENTE_FIN_SEGMENT_1
‘Came entre le point 1 et le point 2

IF CAMSEG_S=2 THEN GOTO ATTENTE_FIN_SEGMENT_2 ‘Came entre le point 2 et le point 3

Voir aussi : CAM_S, CAMNUM_S

9-10-30- CAPTURE1

Syntaxe : CAPTURE1 (<Source>, <N° de l’entrée>, <Front>, < Fenêtre >, <Mini>, <Maxi>, <Intérieur>)

Description : Les instructions CAPTURE1 et CAPTURE 2 sont utilisées pour enregistrer la position courante de l’axe ou de la position du maître.

Avec cette instruction, le variateur attend un front sur l'entrée capture. Quand le front est détecté, la position est stockée dans la variable REGPOS1_S. Le flag REG1_S est alors positionné à vrai.

Type acceptés :<Source> 0 pour position moteur, 1 pour position maître.

<N° de l’entrée> numéro l’entrée sur laquelle on attend le front montant (de 1 à 16).

<Front> 1 sur front montant ou 0 sur front descendant.

<Fenêtre> est vraie alors l'entrée n'est testée que lorsque l’axe est entre les positions <Mini> et <Maxi>.

<Interieur> permet de définir si le test s'effectue à l'intérieur ou à l'extérieur des bornes <Mini> et <Maxi>

<Mini> doit toujours être inférieur à <Maxi>.

Exemple : CAPTURE1 (0, 4, 1, 1, 10, 20,1) 'Capture position du codeur moteur sur front montant de l’entrée 4 lorsque l’axe est entre 10 et 20.

WAIT REG1_S = 1 ‘Attente d’une capture

VR1 = REGPOS1_S ‘VR1 = valeur de la position lors de la capture

Remarque : Temps de capture :

	Entrée normale	Entrée rapide
Filtrage	Filtrage	600 µs
Sans filtrage	150 µs	1 µs

Attention : Il est interdit d’utiliser simultanément la même entrée et le même front sur les fonctions de mouvement déclenché, les captures et les compteurs.

Voir aussi : REG1_S ou REG2_S, REGPOS1_S ou REGPOS2_S

9-10-31- CLEAR – Met à zéro la position de l'axe

Syntaxe : CLEAR

Description : Cette instruction met à zéro la position de l'axe.

Exemple : CLEAR

VR1=POS_S 'Résultat : VR1=0.0

9-10-32- CLEARMASTER - met à zéro la position du codeur maître

Syntaxe : CLEARMASTER

Description : Cette instruction met à zéro la position du codeur maître.

Exemple : CLEARMASTER

9-10-33- COMCOUNTER – Retourne le nombre de trames échangées

Syntaxe : <NB Trames> = COMCOUNTER(X)

Description : Cette instruction retourne le nombre de trames échangées sur la COM sélectionnée : 0 pour le modbus 1 (X1), 1 pour le modbus 2 (X4), 2 pour le CANopen et 3 pour le serveur SDO (incrémenté à chaque réception de requête SDO).

Remarque : Permet de faire une « watchdog » logiciel et vérifier la perte de communication avec un périphérique (IHM, drive ...)

Exemple : TESTCOM :

```
LOADTIMER(VL122)=500 'Charge un temporisation de 5s
WAIT (TIMER(VL122)=0)  'Attente temporisation écoulée
IF OldCounter = COMCOUNTER(1) THEN
    NBErr = NBErr + 1
END IF
OldCounter = COMCOUNTER(1)
IF NBErr >3 GOTO ERRCOM
```

GOTO TESTCOM

9-10-34- CONTINUE – Continue l'exécution d'une tâche

Syntaxe : CONTINUE <n° tâche>

Description : Cette instruction est utilisée pour continuer l'exécution d'une tâche supendue.

Remarques : < n° tâche > doit être le numéro d'une tâche suspendue. Cette fonction n'a pas d'effet sur une tâche stoppée ou en cours d'exécution.

Exemple tâche 1:

```
Wait Inp(9)
        RUN 2
        Begin:
            Wait Inp(9)
            SUSPEND 2
            Wait Inp(8)
            CONTINUE 2
        Goto Begin
```

Voir aussi : RUN, HALT, SUSPEND

9-10-35- COS - Cosinus

Syntaxe : COS(<Expression>)

Types acceptés :Expression : réel

Description : Cette instruction retourne le cosinus de <Expression>.

Remarques : La fonction COS prend un angle et restitue le rapport de deux côtés d'un triangle rectangle. Le rapport est la longueur du côté adjacent à l'angle divisé par la longueur de l'hypoténuse. <Expression>est exprimée en radians.

Exemple : VR0=COS(3.14159)

Voir aussi : SIN, ARCTAN et TAN

9-10-36- COUNTER - Initialise le compteur à une valeur

L'instruction COUNTER(1 ou 2) permet d'écrire une valeur dans les compteurs 1 ou 2.

Syntaxe : COUNTER(1 ou 2) = <Val>

Types acceptés :<Val> : valeur comprise entre 0 et 65535

Description : L'instruction COUNTER(1 ou 2) permet d'écrire une valeur dans les compteurs 1 ou 2.

Exemple : COUNTER(2)=VL1+1000

Attention : Il est interdit d'utiliser simultanément la même entrée et le même front sur les fonctions de mouvement déclenché, les captures et les compteurs.

Voir aussi : SETUPCOUNTER

9-10-37- COUNTER_S – Renvoie la valeur d'un compteur

Syntaxe : <Variable>=COUNTER_S(<n° de compteur>)

Description : L'instruction COUNTER_S permet de lire le compteur.

Type acceptés :<Variable> entier compris entre 0 et 65535

<n° de compteur> numéro de compteur (1 ou 2)

Exemple : VI0 = COUNTER(1)

9-10-38- DAC - Sortie analogique

Syntaxe : DAC = <Expression>

Unité : Expression : Volts

Limites : Expression : de -10 à +10

Types acceptés :Expression : réel

Description : Cette fonction envoie une tension sur la sortie analogique du variateur.

Remarques : La sortie analogique peut également être lue.

Exemple : DAC=5.2

IF ADC(1)>DAC ...

Voir aussi : ADC(1), ADC(2)

9-10-39- DEC - Décélération

Syntaxe 1 : DEC = <Expression>

Syntaxe 2 : <Variable> = DEC

Unité : unité utilisateur par s² (Ex : mm/s², degré/s², tr/s²…)

Types acceptés :Variable, Expression : réel

Description : Cette instruction lit ou modifie la décélération courante en unités par s².

Remarques : La décélération courante peut être lue et modifiée à tout moment.

Exemple : DEC = 500.

VR0 = 10000

DEC = VR0

Voir aussi : ACC, VEL

9-10-40- DEC% - Décélération en pourcentage

Syntaxe : DEC% = <Expression>

Limites : Expression de 0 à 100

Types acceptés : Expression : octet

Description : Cette fonction fixe la décélération courante en pourcentage du paramètre de décélération.

Remarques : La valeur du paramètre de décélération peut être entrée dans l'écran Motion control / Profil de vitesse.

Exemple : DEC% = 10 'Décélération courante 10 %

VB0 = 50

DEC% = 50

Voir aussi : ACC% et VEL%

9-10-41- DELAY – Attente passive

Syntaxe : DELAY <Durée>

Unité : Durée : millisecondes

Types acceptés :Durée : Entier

Description : Cette fonction réalise une attente suivant la durée spécifiée. Elle bloque la tâche et provoque le basculement vers la tâche suivante.

Exemple : DELAY 500 'Délai de 0.5 s.

ou

VI12=500

DELAY VI12

9-10-42- DISABLERECALE– Désactivation du recalage

Syntaxe : DISABLERECALE (<Axe>)

Limites : <Axe> : 0 pour l'axe esclave et 1 pour le maître.

Description : Cette instruction annule le recalage automatique d'un axe sur un capteur

9-10-43- DISPLAY – Afficheur 7 segments

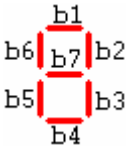
Syntaxe : DISPLAY <Expression>

Types acceptés :Expression : Octet

Description : Cette Instruction fixe l'état de l'afficheur status 7 segments

Remarque : Chaque bit de <Expression> représente un segment. Le dernier bit n'étant pas utilisé.

Exemple : Display 109 ' Equivalent à Display 01101101b ou « 5 »



9-10-44- ENABLERECALE– Fonction de recalage automatique sur capture

Syntaxe : ENABLERECALE (<n° capture>, <Position initiale>, <Accélération>)

Limites : <Position Initiale> : entre 0 et le modulo de l'axe

Types acceptés :<Position Initiale> : Réel

<Accélération> : Réel

Description : Cette instruction recale automatiquement un axe sur un capteur

Remarques : Le paramétrage de la fonction de recalage utilise les paramètres de l'instruction CAPTURE :

<Source> 0 pour position moteur, 1 pour position maître.

<N° de l'entrée> numéro l'entrée sur laquelle on attend le front montant (de 1 à 16).

<Front> 1 sur front montant ou 0 sur front descendant.

<Fenêtre> est vraie alors l'entrée n'est testée que lorsque l'axe est entre les positions <Mini> et <Maxi>.

<Interieur> permet de définir si le test s'effectue à l'intérieur ou à l'extérieur des bornes <Mini> et <Maxi>

<Mini> doit toujours être inférieur à <Maxi>.

L'appel ENABLERECALE annule la fonction CAPTURE dont les paramètres ont été utilisés.

<Position Initiale> indique la position à laquelle se trouve théoriquement le capteur et que l'on mettra dans le compteur. Pour une remise à zéro on indique 0.

<Accélération> indique l'accélération à utiliser pour le recalage

Exemple : …

```
CAPTURE1 (0, 2, 1, 0, 0, 0, 0)      'Capture sur l'esclave et
' sur front montant de l'entrée 2
ENABLERECALE (1, 0, 1000)           'Utilisation des paramètres
' de la capture1
' avec remise de la position à 0
' et avec une accélération de 1000
…
DISABLERECALE (0)
```

9-10-45- EXIT SUB – Sortie d'un sous-programme

Syntaxe : EXIT SUB

Description : Cette instruction permet de sortir d'un sous-programme

Voir aussi : SUB

9-10-46- ENDCAM – Arrêt d'une came

Syntaxe : ENDCAM

Description : Cette fonction permet d'arrêter le mouvement de l'axe à la fin de la came. Elle diffère de la fonction STOP qui met fin au mouvement immédiatement.

Remarques : Attention : Si ENDCAM s'applique à une came qui a été déclarée en mode non mono-coup et enchaînée avec une autre, la came termine son profil et enchaîne sur la suivante.

Voir aussi : STOP

9-10-47- EXP - Exponentiel

Syntaxe : EXP (<Expression>)

Types acceptés :Expression : réel

Description : Cette fonction restitue ***e***(la base des logarithmes naturels) élevée à la puissance <Expression>.

Remarques : La valeur retournée est de type réel.

Exemple : VR2=EXP(2)

Voir aussi : LOG

9-10-48- FEMAX_S – Limite d'erreur de poursuite

Syntaxe : FEMAX_S

Description : Ce flag est mis à 1 lorsque l'erreur de poursuite courante dépasse le seuil du paramètre *erreur de poursuite* accessible à partir du menu Paramètres / Sécurités / Position.

Remarques : Cette fonction est utilisée pour savoir si l'axe est passé en erreur de poursuite Il est nécessaire de traiter ce flag dans une tâche de gestion des défauts si l'instruction SECURITY(0) ou SECURITY(1) a été utilisée.

Remise à 0 du flag :

Si l'entrée logique E1 est configurée en *AUCUNE*, FEMAX_S passe à 0 sur rencontre de l'instruction Axis On dans une tâche basic ou sur front montant du bouton enable à partir de la fenêtre principale du iDPL.

Si l'entrée logique E1 est configurée en *VALIDATION*, FEMAX_S passe à 0 sur front montant de cette entrée.

Si l'entrée logique E1 est configurée en *VALIDATION+iDPL*, FEMAX_S passe à 0 si l'entrée E1 = 1 et exécution dans une tâche basic de l'instruction Axis On.

Exemple : IF FEMAX_S = 1 GOTO Error

GOTO Debut

Error :

Voir aussi : FE_S, SECURITY

9-10-49- FE_S - Erreur de poursuite

Syntaxe : FE_S

Description : Cette fonction retourne une image de l'erreur de poursuite courante.

Remarques : Cette fonction est utilisée pour connaître la valeur courante de l'erreur de poursuite. on peut ainsi vérifier le comportement de la régulation en temps réel.

Exemple : VR1 = FE_S

Voir aussi : FEMAX_S

9-10-50- FILTERMASTER – Permet d'appliquer un filtrage lors de mouvement synchrone

Syntaxe : FILTERMASTER (<Valeur>)

Description : Cette fonction permet d'appliquer différents types de filtrage lors de mouvement synchrone

Les valeurs de <Type> sont :

0 : aucun filtrage, la synchronisation est très rapide mais risque de forts à coup si le ratio est élevé (pour 1 tour maître, l'esclave fait plusieurs tours)

1 : filtrage standard (par défaut)

2 : grâce à une faible constante de temps et un filtrage avancé, la synchronisation garde une réponse assez rapide et diminue fortement les à coup (en particulier pour les ratios élevés)

3 : grâce à une constante de temps élevée et un filtrage avancé, la synchronisation élimine très fortement les à coups mais perd en précision (augmente le temps de réponse)

4 : Filtrage par interpolation pour les forts ratios où la vitesse maître varie peu.

5 : Filtrage avancés par interpolation pour les forts ratios où la vitesse maître varie peu, la synchronisation élimine très fortement les à coups mais perd en précision (augmente le temps de réponse)

9-10-51- FRAC – Partie fractionnelle

Syntaxe : FRAC(<Expression>)

Types acceptés : <Expression> : réel

Description : Cette fonction restitue la partie fractionnelle de <Expression>.

Remarques : Cette fonction restitue une valeur réelle.

Exemple : VR2=3.0214

VR1=FRAC(VR2) 'Résultat VR2=0.0214

Voir aussi : INT

9-10-52- GEARBOX

Syntaxe : GEARBOX(<Numérateur>, <Dénominateur>, <Réversible>))

Description : Cette instruction permet de réaliser un arbre électrique entre un codeur maître et le moteur (axe esclave).

<Numérateur> / <Dénominateur> définit le rapport entre le codeur et l'axe esclave.

<Réversible> indique si la boîte est réversible ou pas, il est mémorisé et sera réutilisé à chaque nouvelle instruction GEARBOXRATIO, STARTGEARBOX

Types acceptés:<Numérateur> est de type réel.

<Dénominateur> est de type réel.

<Réversible> 1 si réversible ou 0 si non réversible

Remarques : Cette instruction est non bloquante pour la tâche (excepté si le buffer de mouvements est plein).Tant que la liaison entre le maître et l'esclave ne sera pas coupée, l'instruction MOVE_S sera égale à 1 (même si l'esclave est arrêté).

Exemples : GEARBOX (1, 2, 1) 'Rapport nominal : ratio 0.5

Voir aussi : GEARBOXRATIO, STARTGEARBOX

9-10-53- GEARBOXRATIO

Syntaxe : GEARBOXRATIO (<Ratio>, <Distance acc. maître>)

Description : Cette instruction permet de modifier le rapport de réduction d'une liaison arbre électrique en cours de mouvement.

Types Acceptés : <Ratio> de type réel : le rapport de l'arbre est défini par <Ratio> × <Numérateur> / <Dénominateur>. <Numérateur> et <Dénominateur> sont les paramètres de l'instruction GEARBOX.

La distance d'accélération maître <Distance acc. maître> correspond à la distance sur laquelle le maître va accélérer.

Remarques : L'instruction est non-bloquante et permet de changer de ratio sans arrêter l'arbre électrique.

Exemple : GEARBOXRATIO (2)

Voir aussi : GEARBOX, STARTGEARBOX

9-10-54- GOTO – Saut à une étiquette

Syntaxe : GOTO <Label>

Description : Réalise un branchement à une étiquette

Remarques : Une étiquette est un nom suivi du caractère ":". l'exécution de cette instruction provoque le basculement vers la tâche suivante.

Exemple : GOTO Begin

...

Begin :

Voir aussi : IF

9-10-55- HALT – Arrêter une tâche

Syntaxe : HALT < n° tâche >

Description : Cette instruction est utilisée pour stopper une tâche en cours d'exécution ou suspendue.

Remarques : Cette fonction n'a pas d'effet sur une tâche déjà arrêtée. Elle n'affecte pas les mouvements en cours ni les buffers de mouvements.

Exemple : Begin :

Wait Inp(8)=On

RUN 2

Wait Inp(8)=Off

HALT 2

Goto Begin

Attention : Après la demande d'arrêt d'un tâche, il est conseillé d'attendre que celle-ci soit terminée avec la commande Wait Status(n° de tâche).

Voir aussi : RUN, SUSPEND, CONTINUE

9-10-56- HOME – Prise d'origine

Syntaxe : HOME(<Type>,[Reference])

Description : Cette fonction force l'axe à se déplacer vers sa position d'origine en utilisant le <Type> de prise d'origine choisi. Cette instruction est bloquante pour la tâche tant que la prise d'origine n'est pas terminée et provoque le basculement vers la tâche suivante. La prise d'origine s'effectue à la vitesse programmée dans l'écran Motion control / Home.

Les valeurs de <Type> sont :

0 : immédiate

1 : Sur Top Z (le variateur n'effectue aucun déplacement mais recalcule sa position par rapport au Top Z d'où une nouvelle position se situant entre +/- ½ tour moteur).

2 : Sur capteur sans dégagement en sens +

3 : Sur capteur avec dégagement en sens +

4 : Sur capteur sans dégagement en sens -

5 : Sur capteur avec dégagement en sens -

6 : Sur capteur et Top Z sans dégagement en sens +

7 : Sur capteur et Top Z avec dégagement en sens +

8 : Sur capteur et Top Z sans dégagement en sens -

9 : Sur capteur et Top Z avec dégagement en sens –

10 : Initialise la position de l'axe avec la position absolue (seulement en mode SinCos ou SSI sinon initialisation à 0)

11 : Remise à 0 de l'erreur de poursuite

12 : Home « relatif » permet de soustraire à la position courant la valeur passé en référence.

[Reference] permet de donner une référence à la prise d'origine

Remarques : Utilisez AXIS Off pour arrêter une prise d'origine en cours. Si <Type> n'est pas spécifié, la valeur est celle indiquée dans l'écran Home du logiciel iDPL.

Exemple : VR0=100

HOME (3,VR0) 'Prise d'origine sur capteur avec dégagement en sens plus et ayant comme référence 100.

Nota : En ajoutant 16 au numéro <Type> de HOME, on effectue alors une prise d'origine qui ne modifie pas la position mais qui stocke le décalage à appliquer dans la variable HOMEPOS_S.

Si le paramètre référence n'est pas entré alors celui-ci est nul.

HOME(2) 'est équivalent à VR0=0 et HOME(2,VR0)

Voir aussi : HOME_S

Attention : Pour les prises d'origine avec capteur, l'entrée 4 doit être déclarée en HOME sinon la prise d'origine est annulée.

9-10-57- HOME_S – Etat de la prise d'origine

Syntaxe : HOME_S

Description : Cette fonction indique l'état de la prise d'origine

Remarques : Cette fonction est utilisée pour savoir si la prise d'origine a été effectuée ou non. Pendant un cycle de prise d'origine, l'indicateur HOME_S est forcé en 0. Dès que le cycle est entièrement terminé, HOME_S passe à 1.

Exemple : IF HOME_S = OFF GOTO Suite

Suite :

Voir aussi : HOME

9-10-58- HOMEMASTER– Prise d'origine sur le maître

Syntaxe : HOMEMASTER (<Type>,[Reference])

Description : Cette fonction permet de faire une prise d'origine forcé de l'axe maître selon le <Type> d'origine choisi. Cette instruction est bloquante pour la tâche tant que la prise d'origine n'est pas terminée et provoque le basculement vers la tâche suivante.

Les valeurs de <Type> sont :

0 : immédiate

1 : Sur Top Z (le variateur attend le top Z sur le codeur maître).

2 : Sur capteur HOME (le variateur attend un front montant sur le capteur HOME)

3 : Sur capteur HOME et Top Z (le variateur attend un niveau haut sur le capteur HOME et un top Z sur le codeur maître)

4 : Initialise la position du maître avec la position absolue (seulement en mode SinCos ou SSI sinon initialisation à 0)

5 : Annule le HOMEMASTER en cours sans modifier le HOMEMASTER_S

[Reference] permet de donner une référence à la prise d'origine

Remarques : Utilisez AXIS Off pour arrêter une prise d'origine en cours. Si <Type> n'est pas spécifié, la valeur est celle indiquée dans l'écran Home du logiciel iDPL.

Exemple : VR0=100

HOMEMASTER (3,VR0) 'Prise d'origine sur capteur avec dégagement en sens plus et ayant comme référence 100.

Nota : En ajoutant 16 au numéro <Type> de HOME, on effectue alors une prise d'origine qui ne modifie pas la position mais qui stocke le décalage à appliquer dans la variable HOMEPOS_S.

Si le paramètre référence n'est pas entré alors celui-ci est nul.

HOMEMASTER (2)

Voir aussi : HOME_S

Attention : Pour les prises d'origine avec capteur, l'entrée 4 doit être déclarée en HOME sinon la prise d'origine est annulée.

9-10-59- HOMEMASTER_S – Etat de la prise d'origine du maître

Syntaxe : HOMEMASTER_S

Description : Cette fonction indique l'état de la prise d'origine du maître

Remarques : Cette fonction est utilisée pour savoir si la prise d'origine a été effectuée ou non. Pendant un cycle de prise d'origine, l'indicateur HOMEMASTER_S est forcé en 0. Dès que le cycle est entièrement terminé, HOMEMASTER_S passe à 1.

Exemple : IF HOMEMASTER_S = OFF GOTO Suite

Suite :

Voir aussi : HOME

9-10-60- ICORRECTION– fonction de compensation

Syntaxe : ICORRECTION (<Dist.maître>, <Dist.esclave>, <Dist. d'accél maître>)

Unités : <Dist.maître>, <Dist.esclave> : unité utilisateur (Ex : mm, degré,…)

<Dist.d'accél> : unité utilisateur/s²

Types acceptés :<Dist.maître>, <Dist.esclave>, <Dist.d'accél> : réel

Description : Cette fonction permet d'appliquer un mouvement de correction sur un axe esclave pendant une distance de l'axe maître.

Remarques : L'esclave devra au préalable être lié à un maître par une fonction d'arbre électrique (GEARBOX), de mouvement synchronisé (MOVS) avant de lancer une compensation. Au mouvement de synchronisation normal de l'esclave, on superpose le mouvement suivant : Pendant que le maître parcourt une « distance maître », on ajoute un déplacement «distance esclave» avec une accélération et une décélération sur une distance maître de «distance d'accél».

Attention : Toute nouvelle correction est ignorée si une correction est déjà en cours ou si la distance maître est nulle.

9-10-61- ICORRECTION_S– Etat de la compensation

Syntaxe : <Variable> = CORRECTION_S

Types acceptés :<Variable> : bit

Description : Cette fonction permet de connaître l'état du cycle de compensation : retourne 1 si ICORRECTION est exécuté sinon renvoie 0.

9-10-62- IF - IF…

Syntaxe 1: IF <Condition> GOTO {<Etiquette>}

Syntaxe 2: IF < Condition > THEN

<Instruction1>

...

END IF

Syntaxe 3: IF < Condition > THEN

<Instruction1>

...

ELSE

<Instruction2>

...

END IF

Description : Permet l'exécution conditionnelle, basée sur l'évaluation d'une expression.

Remarques : Le mot-clé IF commence une structure de contrôle IF.... Il doit apparaître avant toute autre partie de la structure. <Condition> doit être une expression booléenne.

Si <Condition> est vraie alors aller en <Etiquette>.

Exemple : IF VR1=150 GOTO SUITE

9-10-63- INP – Lecture d'une entrée TOR

Syntaxe : INP (<NuméroEntrée>)

Types acceptés :Numéro d'entrées de 1 à 16.

Description : Cette fonction donne l'état d'une entrée logique.

Remarques : <Entrée> doit représenter le numéro de l'entrée logique. Le type de donnée retourné est Bit.

Exemple : VF1 = INP(11)

Voir aussi : INPB, INPW, OUT, OUTB

9-10-64- INPB – Lecture d'un bloc 8 entrées

Syntaxe : INPB (<NuméroBlocEntrées>)

Types acceptés :Numéro d'entrées 1 ou 2.

Description : Cette fonction retourne l'état d'un bloc de 8 entrées logiques.

Remarques : <Entrées> doit représenter le numéro d'un bloc de 8 entrées. Le type donné retourné est octet.

Exemple : VB1=INPB(2)

Voir aussi : INP, INPW, OUT, OUTB

9-10-65- INPW – Lecture des 16 entrées logiques

Syntaxe : INPW

Description : Cette fonction donne l'état du bloc de 16 entrées logiques.

Remarques : Le type de données retourné est entier.

Exemple : VI2=INPW

Voir aussi : INP, INPB, OUT, OUTB

9-10-66- INT – Partie entière

Syntaxe : INT (<Variable>)

Types acceptés :Variable réel

Description : Cette fonction restitue la partie entière d'< Variable >.

Exemple : VR1=25.36

VR2=INT(VR1) 'Résultat : VR2=25

Voir aussi : FRAC

9-10-67- LOADCAM – Permet de charge une came

Syntaxe : LOADCAM (<N°came>, <Absolue>, <Tableau>, <Nombre>, <Mono-coup>, <Réversible>, <Direction>, <GainMaître>, <GainEsclave>, <N°came suivante>, <N°came précédente>)

Description : Cette instruction permet de charger une came dans le variateur.

Limites : <N°came> : numéro de la came (de 1 à 5)

<Absolue> : 1 si came absolue ou 0 si came relative

<Tableau> : indique le 1er point du profile de came (de 0 à 511).

<Nombre> : nombre de ponts du profile de came (de 2à 512).

<Mono-coup> : Définit le rebouclage automatique de la came.

Rentrez la valeur 0 pour une came qui va se reboucler sur son profil jusqu'à ce qu'un arrêt soit demandé, 1 pour une came qui va exécuter son profil une seule fois.

<Réversible> : Indique si l' <Esclave> doit suivre le <Maitre> dans les deux sens.

? Rentrez la valeur 0 pour une came non réversible : si le maître se déplace à l'inverse de son sens normal donné par <Direction>, l'esclave s'arrête ; il repartira lorsque le maître reprendra son sens normal et atteindra la position maître à laquelle l'esclave s'était arrêté.

? Rentrez la valeur 1 pour une came réversible : l'esclave suit son profil de came quel que soit le sens d'avance du maître.

<Direction> : Si la came n'est pas réversible, le sens normal de l'avance du maître doit être indiqué. Rentrez la valeur 0 pour un sens indifférent, 1 pour un sens négatif, 2 pour un sens positif.

<GainMaître> : Cœfficient appliqué sur les positions maître du profil de came (valeur réelle à 1 par défaut).

<GainEsclave>: Cœfficient appliqué sur les positions esclave du profil de came (valeur réelle à 1 par défaut).

<N°came suivante> : Mettez 0 si la came ne doit pas être enchaînée sur une autre came. Dans le cas contraire, rentrez le numéro de la came suivante compris entre 1 et 5.

<N°came précédente> : Mettez 0 si la came n'enchaînera pas sur une came précédente. Dans le cas contraire, rentrez le numéro de la came précédente compris entre 1 et 5.

9-10-68- LOADCAMPOINT – Modification de points d'une came

Syntaxe : LOADCAMPOINT (<N° came>, <N° point>, <Index en FRAM>)

Description : Permet de modifier un point d'une came à partir d'un point en FRAM.

Types acceptés :<N° came> : Numéro de la came cible chargée précédemment (de 1 à 5).

<N° point> : Numéro du point cible de la came (de 1 à nb de points de la came).

<Index en FRAM> : Adresse du point source FRAM (de 0 à 511) à envoyer dans le point cible de la came.

Attention : Cette instruction est bloquante pour la tâche (le chargement d'un nouveau point ne peut se faire si la came se trouve entre les 2 polynômes avant et après le point cible). Cette instruction provoque une erreur DPL E11 si la came cible n'a pas été chargée précédemment.

9-10-69- LOADPARAM – Permet de recharger les paramètres du variateur

Syntaxe : LOADPARAM

Description : Permet de transférer dans la mémoire de travail RAM, les paramètres sauvegardés de la mémoire FLASH.

Voir aussi : SAVEPARAM

9-10-70- LOADVARIABLE - Permet de transférer les variables sauvegardés

Syntaxe : LOADVARIABLE

Description : Permet de transférer dans la mémoire de travail, les variables VR0 à VR63 et VL0 à VL63 sauvegardés de la mémoire FLASH.

Voir aussi : SAVEVARIABLE

9-10-71- LOADTIMER - Charge une temporisation dans une variable

Syntaxe : LOADTIMER(<VL n°XX>)=<Val>

Types acceptés :Val : entier long

Description : L'instruction LOADTIMER peut être utilisée pour établir des attentes actives. Elle stocke dans la variable VLXX, la somme de Time + <Val>

Remarques : Il est possible d'utiliser simultanément 256 timers.

Exemple : LOADTIMER(VL129)=3000 'Charge une temporisation de 3000ms dans la variable VL129

Voir aussi : TIMER

9-10-72- LOG - Logarithme

Syntaxe : LOG (<Expression>)

Types acceptés :Expression : réel

Description : Retourne le logarithme naturel de <Expression>

Exemple : VR0=LOG(VR1)

Voir aussi : EXP

9-10-73- LOOP – Mode virtuel

Syntaxe : LOOP ON/OFF

Description : Cette fonction passe l'axe en mode virtuel et permet de tester un programme sans codeur ni moteur. Dans ce mode, ne pas brancher la puissance sur le connecteur X10

La fonction LOOP ON permet d'ignorer les erreurs E2, E7 et E8.

9-10-74- MASTEROFFSET – Décale dynamiquement la position du maître

Syntaxe : MASTEROFFSET (<Offset>, <Accélération>)

Description : Cette instruction décale dynamiquement la position du maître utilisée par la came absolue

Limites : <Offset> : entre 0 et le modulo du maître

Types acceptés :<Offset> : Réel

<Accélération> : Réel

Remarques : <Offset> : Valeur de l'offset à appliquer

<Accélération> accélération utilisée pour appliquer l'offset (dans l'unité du maître).

Attention : Le décalage est appliqué directement si le mouvement synchro n'est pas en cours ou si l'axe n'est pas embrayé.

9-10-75- MERGE – définit l'enchaînement

Syntaxe : MERGE ON | OFF

Description : Cette instruction est utilisée pour activer ou désactiver l'enchaînement des mouvements consécutifs.

Exemple :

```
MERGE ON
TRAJA(1000,500)   'Mouvements enchaînés sans
TRAJA(1500,200)   'passage par une vitesse nulle
MERGE OFF
TRAJA(1800,700)   'passage par une vitesse nulle à la position 1500
```

9-10-76- MOD - Modulo

Syntaxe : <Expression1> MOD <Expression2>

Types acceptés :Expression1, Expression2 : Octet, Entier, Entier long

Description : Cet opérateur restitue le reste d'une division entière.

Exemple : VI10=5

VI10=VI10 MOD 2 'Résultat : VI10=1

9-10-77- MOVA – Mouvement absolu

Syntaxe : MOVA = <Distance>

Unité : Distance : unité utilisateur (Ex : mm, degré,…)

Types acceptés :Distance : réel

Description : Déplace l'axe à une position absolue. L'exécution de l'instruction provoque le basculement vers la tâche suivante.

Remarques : La tâche attend la fin du mouvement (condition MOVE_S=0) avant d'exécuter la prochaine instruction. L'axe utilise les valeurs courantes de vitesse, d'accélération et de décélération.

Exemple : MOVA = 1200.00

Voir aussi : MOVR, STTA, STTR, STTI et MOVE_S

9-10-78- MOVE_S – Etat du mouvement

Syntaxe : MOVE_S

Types acceptés :Bit

Description : Cette fonction indique si l'axe est en mouvement (trajectoire simple ou mouvement synchronisé).

Remarques : Si l'axe est en mode non asservi (AXIS OFF), l'instruction MOVE_S = 0. Si l'axe est en mode asservi, MOVE_S est égale à 0 si les 4 points suivants sont vrais :

Le mouvement de positionnement (STTA, TRAJ…) courant est terminé (trajectoire théorique terminée).

L'erreur de poursuite est à l'intérieur de la fenêtre de positionnement (+/- la valeur entrée dans le menu Paramètres / Sécurités / Position).

Le buffer de mouvement de positionnement (STTA, TRAJ…) est vide.

Dans le cas d'un axe esclave lié par une synchronisation (arbre électrique, mouvement synchronisé, mouvement came) : le lien doit avoir été coupé.

Si l'un de ces points est faux, l'instruction MOVE_S retourne la valeur 1.

Exemple : STTA = VR10

WAIT MOVE_S = OFF ' Attente que l'axe soit arrêté

Remarque : En mode VIRTUALMASTER, MOVE_S est égale à 0 si les 3 points suivants sont vrais :

L'erreur de poursuite est à l'intérieur de la fenêtre de positionnement (+/- la valeur entrée dans le menu Paramètres / Sécurités / Position).

Le buffer de mouvement synchronisé est vide.

Dans le cas d'un axe esclave lié par une synchronisation (arbre électrique, mouvement synchronisé, mouvement came) : le lien doit avoir été coupé.

9-10-79- MOVEMASTER_S – Indique si un mouvement est en cours lorsqu'on est en maître virtuel

Syntaxe : MOVEMASTER_S

Types acceptés :Bit

Description : MOVEMASTER_S est égal à 0 si les 3 points suivants sont vrais :

On est en maître virtuel.

Le mouvement de positionnement (STTA, TRAJ…) courant est terminé (trajectoire théorique terminée).

Le buffer de mouvement de positionnement (STTA, TRAJ…) est vide.

Exemple : VIRTUALMASTER ON

STTA = VR10

WAIT MOVEMASTER_S = OFF ‘ Attente que la trajectoire sur

‘ le maître virtuel soit arrêté

9-10-80- MOVR – Mouvement relatif

Syntaxe : MOVR = <Distance>

Types acceptés :Distance : réel

Description : Déplace l'axe à une position relative. L'exécution de l'instruction provoque le basculement vers la tâche suivante.

Remarques : La tâche attend la fin du mouvement (condition MOVE_S=0) avant d'exécuter la prochaine instruction. L'axe utilise les valeurs courantes de vitesse, d'accélération et de décélération.

Exemple : MOVR = VR1

Voir aussi : MOVA, STTA, STTR, STTI, MOVE_S

9-10-81- MOVS - permet d'effectuer une synchronisation entre un axe esclave et un maître.

Description : Cette instruction est non bloquante pour la tâche (excepté si le buffer de mouvements est plein).

Syntaxe : MOVS (<Dist. maître>, <Dist . esclave>,<Dist. d'accél.> , <Dist. de décél.>)

Exemple : MOVS (20, 10, 0, 0) 'pour un déplacement relatif de 20 unités

'sur le maître, l'esclave se déplace de 10

9-10-82- NEXTTASK

Syntaxe : NEXTTASK

Description : Instruction permettant de faire un basculement immédiat vers la tâche suivante.

9-10-83- NOT – Opérateur complément

Syntaxe : NOT(<Expression>)

Types acceptés :Expression : Bit, Octet, Entier

Description : La fonction NOT retourne le complément.

Exemple : VB1=15

VB2=NOT VB1 'Résultat VI2=140

Voir aussi : AND, OR, XOR

9-10-84- OR - Opérateur ou

Syntaxe : <Expression1> OR <Expression2>

Types acceptés :Expression1, Expression2 : Bit, Octet, Entier

Description : Cette fonction effectue un OU binaire entre deux expressions.

Remarques : <Expression1> et <Expression2> doivent être du même type. Cette fonction restitue le même type de donnée que ses arguments

Exemple : VI12=VI12 OR 000FFh

Voir aussi : AND, NOT, XOR et IF

9-10-85- ORDER – Numéro d'ordre du mouvement

Syntaxe 1 : ORDER = <Valeur>

Syntaxe 2 : ORDER

Types acceptés :Valeur : entre 0 et 65535

Description : Cette instruction fixe le numéro d'ordre+1 du prochain mouvement ou lit le numéro d'ordre du dernier mouvement déposé.

Remarques : Cette instruction peut être utilisée avec la fonction ORDER_S.

Exemple : ORDER = 0

STTA = 50

VB1 = ORDER 'Résultat : VB1=1

Voir aussi : ORDER_S

9-10-86- ORDER_S – Numéro d'ordre courant

Syntaxe : ORDER_S

Types acceptés :Entier

Description : Cette fonction retourne le numéro du mouvement en cours d'exécution.

Remarques : Cette fonction peut être utilisée pour connaître l'état du mouvement.

Exemple : ORDER=0

STTA = 50

STTA = 100

STTA = 50

IF ORDER_S=2 ...'Le second mouvement est commencé.

Voir aussi : ORDER

9-10-87- OUT – Ecriture d'une sortie

Syntaxe : OUT (<NumSortie>) = <Expression>

Types acceptés :Expression : Bit

Description : Cette fonction envoie un état logique à une sortie TOR.

Remarques : <Sortie> doit représenter le numéro d'une sortie de 1 à 10

Exemple : OUT(10) = ON

Voir aussi : INP, INPB, INPW, OUTB

9-10-88- OUTB – Ecriture d'un bloc de 8 sorties

Syntaxe : OUTB (<NuméroBlocSorties>) = <Expression>

Types acceptés :<Expression> : de type octet

(<NuméroBlocSorties> : 1 ou 2

Description : Cette fonction envoie des états logiques à un bloc de 8 sorties TOR.

Exemple : OUTB(1)=15

Voir aussi : INP, INPB, INPW, OUT

9-10-89- POS – Position à atteindre

Syntaxe 1 : POS = <Expression>

Syntaxe 2 : POS

Types acceptés :Expression : réel

Description : Cette fonction retourne ou fixe la position à atteindre dans l'unité choisie.

Remarques : Cette fonction est utilisée pour changer la position finale en cours de mouvement. La position peut être modifiée à tout moment.

Exemple : STTA = 5000 'Départ de l'axe

WAIT INP(10) = On 'Attente Cellule

```
POS = POS_S+50.          'Arrêt 50 mm après le capteur
WAIT MOVE_S = OFF        'Attente arrêt de l'axe
```

Voir aussi : ACC, DEC, VEL

9-10-90- POS_S – Position réelle

Syntaxe : <Expression> = POS_S

Types acceptés :Expression : réel

Description : Cette fonction retourne la position réelle de l'axe.

Remarques : On peut ainsi obtenir l'image en temps réel de la position de l'axe.

Exemple :

```
STTA = 100               'Départ de l'axe
OUT(5) = 1               ‘Activation sortie n°5
BOUCLE :
VR1=POS_S
IF VR1<50 GOTO BOUCLE
OUT(5) = 0               ‘Désactivation de la sortie n°5
```

Voir aussi : VEL_S

9-10-91- POSMASTER_S – Position réelle du maître

Syntaxe : <Expression> = POSMASTER_S

Types acceptés :Expression : réel

Description : Cette fonction retourne la position réelle de l'axe maître.

Remarques : On peut ainsi obtenir l'image en temps réel de la position de l'axe maître.

Exemple :

```
STTA = 100               'Départ de l'axe
OUT(5) = 1               ‘Activation sortie n°5
BOUCLE :
VR1=POSMASTER_S
IF VR1<50 GOTO BOUCLE
STOP                     ‘Arrêt de l’axe par rapport à l’axe maître
```

```
OUT(5) = 0          'Désactivation de la sortie n°5
```

Voir aussi : VEL_S

9-10-92- PROG .. END PROG – Début d'un programme

Syntaxe : PROG

Description : Ce mot-clé commence un bloc de programme principal. Il est également utilisé pour identifier la fin d'un bloc de programme principal lorsqu'il est précédé de END.

Remarques : Un et seulement un bloc PROG - END PROG doit être défini dans un programme

Exemple :

```
PROG

    ...

END PROG
```

9-10-93- READCAM – Permet de lire un point de came

Syntaxe : <VRx>=ReadCam(<Index>, <Sous index>)

Description : Cette fonction permet de lire un point de came à partir de la mémoire FRAM

Limites : <Index> de 0 à 511, numéro du point de came en FRAM

<Sous index> de 0 à 3, paramètre du point de came :

- 0 pour la position de maître
- 1 pour la position de l'esclave
- 2 pour la tangente maître
- 3 pour la tangente esclave

<VRx> de VR0 à VR255

Voir aussi : WRITECAM

9-10-94- READI - Lecture d'un entier en FRAM

Lecture : <VIn> = READI (<Adresse>)

Limites : <Adresse> : de 0 à 4095

n de 0 à 255

9-10-95- READL -Lecture d'un entier long en FRAM

Lecture : <VLn> = READL (<Adresse>)

Limites : <Adresse> : de 0 à 4094

n de 0 à 255

Attention : La lecture et l'écriture d'un entier long utilise 2 adresses mémoires consécutives (adresse et adresse+1).

9-10-96- READR - Lecture d'un réel en FRAM

Lecture : <VRn> = READR (<Adresse>)

Limites : <Adresse> : de 0 à 4094

n de 0 à 255

Attention : La lecture et l'écriture d'un réel utilise 2 adresses mémoires consécutives (adresse et adresse+1).

9-10-97- READPARAM - Lecture d'un paramètre

Syntaxe : <Variable> = READPARAM (<Index>, <Sub-Index>)

Types acceptés :<Variable> du type entier long

<Index> de type entier

<Sub-Index> de type octet

Description : Cette fonction permet de lire via le bus CANopen, les paramètres du variateur.

Exemple : VL0 = READPARAM(8448,1) 'Renvoie le numéro du défaut du variateur

9-10-98- REG1 S

Syntaxe : <VFx>=REG1_S

Description : Cette fonction indique si une capture de position a été effectuée.

Remarques : La valeur retournée n'est vraie qu'une fois par capture. REG1_S est remis automatiquement à 0 sur une opération de lecture et lorsqu'il

vaut 1. Sur une relance d'une autre capture et si REG1_S vaut 1 alors REG1_S est mis à 0.

Exemple : CAPTURE1(0,4,1,10,20,1) 'Capture de la position du codeur moteur

sur front montant de l'entrée 4 lorsque l'axe est entre 10 et 20.

WAIT REG1_S = 1 'Attente d'une capture

VR1 = REGPOS1_S 'VR1 = valeur de la position lors de la capture

Voir aussi : CAPTURE1 ou CAPTURE2, REGPOS1_S ou REGPOS2_S

9-10-99- REGPOS1 S

Syntaxe : <VR XX>=REGPOS1_S

Description : Cette fonction retourne la dernière position capturée sur l'axe par l'exécution de l'instruction CAPTURE1.

Exemple : CAPTURE1(0,4,On,10,20,On) 'Capture de la position du codeur moteur sur front montant de l'entrée 4 lorsque l'axe est entre 10 et 20.

WAIT REG1_S = ON'Attente d'une capture

VR1 = REGPOS1_S 'VR1 = valeur de la position lors de la capture

Voir aussi : CAPTURE1 ou CAPTURE2, REG1_S ou REG2_S

9-10-100- REPEAT ... UNTIL – Répétition d'une boucle

Description : L'instruction REPEAT permet l'exécution répétée d'une ou plusieurs instructions selon la valeur d'une expression.

Syntaxe : REPEAT

<Instructions>

…

UNTIL <Expression>

<Expression> doit être une valeur de type bit, si <Expression> est VRAIE avant la structure REPEAT, la boucle est effectuée une fois. <Instructions> sont exécutées jusqu'à ce que <Expression> soit vraie.

Exemple : VEL% = 100 ' Vitesse rapide

STTA = 2000 ' Start absolu en position 2000

REPEAT

VR0 = POS_S

```
IF VR0>1000 THEN
VEL% =50                ' Vitesse lente à la moitié
END IF                  ' de la distance
UNTIL NOT MOVE_S        ' reboucler jusqu'à ce que
                        ' le moteur soit arrêté
```

9-10-101- RESTART – Redémarrage du système

Syntaxe : RESTART

Description : Redémarre le système de la même manière qu'une mise sous tension.

9-10-102- RUN – Lance une tâche

Syntaxe : RUN < n° tâche >

Description : Cette instruction est utilisée pour lancer une tâche à l'arrêt (ex : tâche déclarée en démarrage manuel).

Remarques : Cette fonction n'a pas d'effet sur une tâche suspendue ou déjà lancée.

Exemple :

```
Debut:
Wait Inp(11)=On
RUN  3
Wait Inp(11)=Off
HALT 3
Goto Debut
```

Voir aussi : CONTINUE, HALT, SUSPEND

9-10-103- SAVEPARAM - Permet de sauvegarder les paramètres du variateur

Syntaxe : SAVEPARAM

Description : Les paramètres du variateurs en RAM EXTERNE sont sauvegardés en mémoire XFLASH.

Remarque : La FLASH a une durée de vie de 5000 cycles d'écriture.

Voir aussi : LOADPARAM

Attention : Consulter notre service technique avant l'utilisation de cette instruction sous peine de dégradation prématurée de la mémoire FLASH

L'utilisation des instructions SAVEPARAM et SAVEVARIABLE fausse la base de temps et provoque l'arrêt de l'envoie de la position CAN.

9-10-104- SAVEVARIABLE – Permet de sauvegarder les variables

Syntaxe : SAVEVARIABLE

Description : Les variables en RAM VR0 à VR63, VL0 à VL63 sont sauvegardées en mémoire FLASH.

Le variateur passe automatiquement en AXIS OFF

Remarque : La FLASH a une durée de vie de 5000 cycle d'écriture.

Voir aussi : LOADVARIALBE

Attention : Consulter notre service technique avant l'utilisation de cette instruction sous peine de dégradation prématurée de la mémoire FLASH

L'utilisation des instructions SAVEPARAM et SAVEVARIABLE fausse la base de temps et provoque l'arrêt de l'envoie de la position CAN.

9-10-105- SECURITY – Définit les actions de sécurités

Syntaxe : SECURITY(<Niveau>)

Description : Cette instruction est utilisée pour définir comment le système doit réagir si une erreur de poursuite sur l'axe est détectée. <Niveau> détermine le niveau de sécurité. Les valeurs par défaut à la mise sous tension sont SECURITY(2)

Niveau	Err. 12 sur afficheur	Flag Femax = 1	Etat de l'instruction Axis_S	S1 (ready) = 0
0	Non	1	Axis_s = On	1
1	Non	1	Axis_s = Off	1
2	Oui	1	Axis_s = Off	0

Remarques : Si l'instruction SECURITY est utilisée, le niveau de sécurité peut être diminué suivant l'écriture du programme. Il est conseillé de ne pas utiliser cette instruction.

Exemple : SECURITY(0) ' L'axe reste asservi en cas d'erreur de poursuite

Nota : Le flag Femax_S est remis à 0 lorsque l'on repasse en mode asservi (Axis On).

9-10-106- SETUPCOUNTER – Configure un compteur

Syntaxe : SETUPCOUNTER(<1 ou 2>, <Num Entrées>, <Filtre>)

Types acceptés :<Filtre> : bit

Description : Cette instruction permet de configurer les compteurs 1 ou 2

Remarques : <Num Entrée> : Numéro de l'entrée de 1 à 16

<Filtre> : Validation du filtre : 0 pour sans filtrage, 1 pour avec filtrage.

Voir aussi : COUNTER

Attention : Si le filtre n'est pas activé, la fréquence maxi est de 781Hz soit 1,24 ms sinon il dépend du paramètre Filtrage dans Paramètres / Entrées Sorties Digitales .

9-10-107- SGN - Signe

Syntaxe : SGN (<Expression>)

Types acceptés :Expression : Entier long, réel

Description : Cette fonction retourne un réel égal à -1 pour les nombres négatifs, 1 pour les nombres positifs et 0 pour les nombres nuls.

Exmple : VR0=SGN(10) 'Résultat : VR0=1

9-10-108- SIN - Sinus

Syntaxe : SIN (<Expression>)

Types acceptés : Expression : réel

Description : Cette instruction retourne le sinus de <Expression>. <Expression>est exprimée en radians.

Voir aussi : COS, ARCTAN, TAN

9-10-109- SLAVEOFFSET – Décale dynamiquement la position de l'esclave

Syntaxe : SLAVEOFFSET (< Offset>, <Accélération>)

Description : Cette instruction décale dynamiquement la position de l'esclave

Limites : <Offset> : entre 0 et le modulo de l'esclave

Types acceptés :<Offset> : Réel

<Accélération> : Réel

Remarques : <Offset> : Valeur de l'offset à appliquer

<Accélération> accélération utilisée pour appliquer l'offset (dans l'unité de l'esclave).

Attention : Le décalage est appliqué directement si le mouvement synchro n'est pas en cours ou si l'axe n'est pas embrayé.

9-10-110- SQR - Racine carrée

Syntaxe : SQR (<Expression>)

Types acceptés : Expression : réel

Description : Cette fonction retourne la racine carrée de <Expression>.

Exemple : VR0=SQR(2)

9-10-111- SSTOP – Arrêt d'un axe

Syntaxe : SSTOP

Description : Cette fonction stoppe l'axe avec la décélération courante. La fonction n'est pas bloquante pour la tâche.

Remarques : Si l'axe exécute un mouvement de synchronisation, alors l'axe s'arrête.

L'instruction SSTOP vide le buffer de mouvement et stoppe l'axe en utilisant la décélération courante. Cette instruction n'est pas bloquante et n'attend pas que MOVE_S soit égal à 0.

En mode maître virtuel, SSTOP n'arrête pas les mouvements de positionnement (STTA, TRAJA …)

Exemple : SSTOP

Voir aussi : STTA, STTR, STTI, GEARBOX, CAMBOX

9-10-112- SSTOPMASTER – Arrête le mouvement du maître virtuel sans attente

Syntaxe : SSTOPMASTER

Description : Cette fonction arrête le mouvement du maître virtuel. La fonction n'est pas bloquante pour la tâche.

Remarques : L'instruction SSTOPMASTER vide le buffer de mouvement du maître et stoppe le maître virtuel en utilisant la décélération courante. Cette instruction n'est pas bloquante et n'attend pas que MOVEMASTER_S soit égal à 0.

Exemple :
```
VIRTUALMASTER ON
        MOVS (1, 1, 0, 0)
STTA = 10
…
SSTOPMASTER      ' Demande d'arrêt du maître
WAIT MOVEMASTER_S = 0 ' Attente fin de mouvement du maître
STTA = 10   ' le maître redémarre et l'axe recommence à tourner
```

9-10-113- STARTCAM – Exécute une came

Syntaxe : STARTCAM (<N°came>)

Limites : <N°came> : de 1 à 5

Types acceptés : <N°came> : Octet

Description : Cette instruction lance l'exécution d'une came

Voir aussi : LOADCAM

9-10-114- STARTCAMBOX – Lance une boîte à cames

Syntaxe : STARTCAMBOX(<Num boîte>)

Description : Cette instruction lance une boîte à cames précédemment définie.

Remarques : Si la boîte à cames n'est pas définie, la fonction n'a pas d'effet.<Num boîte> est le numéro utilisé dans la fonction CAMBOX.

Exemple : STARTCAMBOX(1)

Voir aussi : CAMBOX

9-10-115- STARTGEARBOX - Lance l'arbre électrique

Syntaxe : STARTGEARBOX(<Distance acc. Maître>)

Description : Cette instruction permet de lancer un arbre électrique suivant une distance d'accélération et un rapport défini précédemment par GEARBOX et GEARBOXRATIO.

Types Acceptés : <Distance acc. Maître> de type réel

Exemples : STARTGEARBOX(20) 'Lance l'arbre électrique avec une phase

… 'd'accélération de 20 unité du maître

Voir aussi : GEARBOX, GEARBOXRATIO

9-10-116- STATUS – Etat d'une tâche

Syntaxe : STATUS (<n° tâche>)

Description : Cette fonction retourne l'état d'une tâche

Remarques : Les valeurs possibles sont :

0 : La tâche est stoppée

1 : La tâche est suspendue

2 : La tâche est en cours d'exécution

Exemple : Run 2

Wait Status(2)=0

9-10-117- STOP - Arrêt d'un axe

Syntaxe : STOP

Description : Cette fonction stoppe l'axe esclave avec la décélération courante. La fonction est bloquante tant que l'axe n'est pas arrêté.

Remarques : Si l'axe exécute un mouvement de synchronisation, alors l'axe s'arrête.

L'instruction STOP vide le buffer de mouvement et stoppe l'axe en utilisant la décélération courante. Cette instruction est bloquante tant que MOVE_S est différent de 0.

Exemple : STOP

Attention : Après la demande d'arrêt d'une tâche, il est conseillé d'attendre que celle-ci soit terminée avec la commande Wait Status(n° de tâche).

En mode maître virtuel, STOP n'arrête pas les mouvements de positionnement (STTA, TRAJA …)

Voir aussi : STTA, STTR, STTI, GEARBOX

9-10-118- STOPCAMBOX – Arrête une boîte à cames

Syntaxe : STOPCAMBOX(<Num boîte>)

Description : Cette instruction arrête une boîte à cames précédemment définie.

Remarques : <Num boîte> est le numéro utilisé dans la fonction CAMBOX. Cette fonction ne détruit pas la boîte.

Exemple : STOPCAMBOX(1)

Voir aussi : CAMBOX, CAMBOXSEG, STARTCAMBOX

9-10-119- STOPS - permet d'arrêter l'instruction MOVS

Description : Lorsque l'axe maître atteint <Pos.maître>, l'axe esclave commencera à décélérer pour atteindre <Pos.esclave>.

Syntaxe : STOPS (<Pos.maître >, <Pos. esclave)

<Pos.maître > du type réel, position dans l'unité du maître.

<Pos.esclave > du type réel, position dans l'unité de l'esclave.

Exemple : STOPS (20, 105) 'Quand l'axe maître aura atteint la position 20,

' l'esclave décélérera pour atteindre la position 105 sur l'esclave.

Attention : L'appel de l'instruction STOPS remet le flag STOPS_S à zéro.

9-10-120- STOPS_S - état du mouvement synchronisé

Description : Ne sert que si l'instruction STOPS a été appellée précédemment. Ce flag indique si la position esclave donné par l'instruction STOPS n'a pu être atteinte. Il est remis à zéro après chaque lecture.

Syntaxe : VF0 = STOPS_S

Description: Retourne 1 si :

1er cas la position esclave demandé par l'instruction STOPS n'est pas réalisable (ex : la position esclave demandée par STOPS est déjà dépassée.)

2ième cas : si la vitesse esclave est nulle (en phase de plateau).

Sinon retourne 0

Exemple : MOVS (20, 10, 0, 0)

...

STOPS (20, 105)

WAIT MOVE_S=0

IF STOPS_S=1 GOTO ERRSTOPS

9-10-121- STOPMASTER – Arrête le mouvement du maître virtuel

Syntaxe : STOPMASTER

Description : Cette fonction arrête le mouvement du maître virtuel. La fonction est bloquante tant que le mouvement n'est pas arrêté.

Remarques : L'instruction STOPMASTER vide le buffer de mouvement du maître et stoppe le maître virtuel en utilisant la décélération courante. Cette instruction est bloquante tant que MOVEMASTER_S est différent de 0.

Exemple : VIRTUALMASTER ON

MOVS (1, 1, 0, 0)

STTA = 10

…

STOPMASTER ‘ le maître s'arrête, l'axe ne tourne plus

‘ mais le mouvement synchro est toujours actif

STTA = 10 ‘ le maître démarre et l'axe recommence à tourner

9-10-122- STTA – Lance un mouvement absolu

Syntaxe : STTA = <Distance>

Types acceptés :Distance : réel

Description : Lance un mouvement à une position absolue.

Remarques : Le système n'attend pas la fin du mouvement (MOVE_S=0) et exécute la prochaine instruction. L'axe utilise la vitesse, l'accélération et la décélération courante

Exemple : STTA = 1200.00

WAIT MOVE_S = OFF

Voir aussi : MOVA, MOVR, STTR, STTI

9-10-123- STTI – Lance un mouvement infini

Syntaxe : STTI + ou -

Description : Lance un mouvement infini.

Remarques : Le système n'attend pas la fin du mouvement et exécute la prochaine instruction. Vous devez utiliser l'instruction STOP ou SSTOP pour arrêter le mouvement. L'axe utilise la vitesse et l'accélération courante.

Exemple : STTI + ' commence un mouvement infini
‘ dans la direction positive.

Voir aussi : MOVA, MOVR, STTA, STTR, STOP

9-10-124- STTR – Lance un mouvement relatif

Syntaxe : STTR = <Distance>

Types acceptés :Distance : réel

Description : Lance un mouvement relatif.

Remarques : Le système n'attend pas la fin d'un mouvement (MOVE_S=0) avant d'exécuter la prochaine instruction. L'axe utilise la vitesse, l'accélération et la décélération courante

Exemple : VR0 = 420

STTR = VR0

Voir aussi : MOVA, MOVR, STTA, STTI

9-10-125- SUB .. END SUB – Sous-programme

Syntaxe : SUB <Nom>

Description : Ce mot-clé commence un bloc de sous-programme et est aussi utilisé pour définir la fin d'un bloc de sous-programme quand il est précédé de END.

Remarques : Les blocs SUB - END SUB doivent être en dehors d'un bloc PROG – END PROG.

Exemple : SUB Move

...

END SUB

9-10-126- SUSPEND – Suspend une tâche

Syntaxe : SUSPEND < n° tâche >

Description : Cette instruction suspend une tâche en cours d'exécution.

Remarques : Cette instruction n'a pas d'effet sur les tâches stoppées. Les mouvements présents dans le buffer de mouvement continuent à s'exécuter.

Exemple :

```
Wait Inp(12)
RUN  4
Begin:
Wait Inp(12)
SUSPEND 4
Wait Inp(12)
CONTINUE 4
Goto Begin
```

Voir aussi : RUN, CONTINUE, HALT

9-10-127- TAN - Tangente

Syntaxe : TAN (<Expression>)

Types acceptés :Expression : réel

Description : Cette instruction retourne la tangente de <Expression>. <Expression> est un angle exprimé en radians.

Remarques : La fonction TAN prend un angle et restitue le rapport de deux côtés d'un triangle rectangle. Le rapport est la longueur du côté opposé d'un angle divisée par la longueur du côté adjacent à l'angle.

Exemple : VR=TAN(3.14)

Voir aussi : SIN, ARCTAN,TAN

9-10-128- TIME - Base de temps étendue

Syntaxe : <VLx> = TIME

Description : La variable système TIME peut être utilisée pour établir des attentes actives. C'est un entier long qui représente le millième de secondes écoulées depuis la dernière mise sous tension.

Exemple : VL2=TIME

```
VL2= VL2 + 5000      'Charge une temporisation de 5000ms
ATTENTE:
VL3 = TIME
 IF VL3<VL2 GOTO ATTENTE
```

Attention : La fonction TIME ne fonctionne pas dans un test

9-10-129- TIMER – Comparaison une variable à Time

Syntaxe : TIMER(<VL XX>)

Description : Cette instruction compare la variable système Time et le contenu de la variable VLXX :

TIMER(VLXX) =1 si Time<=VLXX (temporisation en cours).

TIMER(VLXX)=0 si Time>VLXX (temporisation écoulée).

Types acceptés :VL XX : Variable du type entier long

Exemple : LOADTIMER(VL122)=3000 'Charge un temporisation de 3s

WAIT (TIMER(VL122)=0) 'Attente temporisation écoulée

9-10-130- TRAJA – Trajectoire absolue

Syntaxe : TRAJA (<Position>,<Vitesse>)

Types acceptés :<Position> du type réel

<Vitesse> du type réel

Description : Cette instruction effectue une trajectoire complexe. L'exécution de cette tâche provoque le basculement vers la tâche suivante.

Remarques : L'axe utilise l'accélération et la décélération courante.

Exemple :

```
MERGE On                  'Passage en petite vitesse
TRAJA (1000.00, VR0)      'à la position 1000,
TRAJA (1500.00, VR1)      ' sans arrêt de l'axe
MERGE Off
```

Voir aussi : STTA, MERGE, TRAJR

9-10-131- TRAJR – Trajectoire relative

Syntaxe : TRAJR (<Position>,<Vitesse>)

Types acceptés :<Position> du type réel

<Vitesse> du type réel

Description : Cette instruction effectue une trajectoire complexe. L'exécution de cette tâche provoque le basculement vers la tâche suivante.

Remarques : L'axe utilise l'accélération et la décélération courante.

Exemple :

```
MERGE On                    'Passage en petite vitesse
TRAJR (200.00, VR0)
TRAJR (1000.00, VR0)        'à la position 1200,
TRAJR (1500.00, VR1)        ' sans arrêt de l'axe
MERGE Off
```

Voir aussi : STTR, MERGE, TRAJA

9-10-132- TRIGGERC - Mouvement déclenché sur entrée capture

Description : Cette instruction indique que le prochain mouvement sera déclenché sur un numéro de capture.

Syntaxe : TRIGGERC (<N° de capture>)

< N° de capture > de 1 à 2.

Exemple :

```
STTA =50 'mouvement absolue en 50 sans attente de fin de
mouvement
…
CAPTURE1(0,4,On,10,20,On)        'Capture position sur front
montant
'de l'entrée 4 lorsque l'axe du
'moteur est entre 10 et 20.
TRIGGERC (1)
STTA =300   'mouvement absolue en 300
            ' déclenché sur déclenchement de la capture 1.
```

Attention : l'exécution de l'instruction TRIGGERC, annule la capture, il est donc possible de recharger une nouvelle capture

Le TRIGGERC avec les entrées 3, 4, 15, 16 fonctionnent comme avec des entrées standards.

9-10-133- TRIGGERI – Mouvement déclenché sur entrée

Description : Cette instruction indique que le prochain mouvement sera déclenché sur changement d'état d'une entrée.

Syntaxe : TRIGGERI (<N° d'entrée>, <Front>)

< N° d'entrée > de 1 à 16.

< Front > 0 front descendant, 1 front montant.

Exemple : STTA =50 'mouvement absolue en 50 sans attente de fin de mouvement

…

TRIGGERI (7,1)

STTA =300 'mouvement absolue en 300

' déclenché sur front montant de l'entrée 7

Attention : Il est interdit d'utiliser simultanément la même entrée et le même front sur les fonctions de mouvement déclenché, les captures et les compteurs.

9-10-134- TRIGGERP – Mouvement déclenché sur position maître

Description : Cette instruction indique que le prochain mouvement sera déclenché sur le passage du maître à une position.

Syntaxe : TRIGGERP (<Pos. Maître>, <Sens>)

<Pos.maître > du type réel, position dans l'unité du maître.

<Sens > 0 indifférent, 1 sens négatif, 2 sens positif.

Exemple : STTA =50 'mouvement absolue en 50 sans attente de fin de mouvement

…

TRIGGERP (200,2)

STTA =300 'mouvement absolue en 300

‘ déclenché lorsque la position du maître atteindra 200

‘ dans le sens positif

9-10-135- TRIGGERR – Annule le mouvement déclenché

Cette instruction annule le mouvement déclenché sans aucune condition.

Cette instruction doit être utilisée dans une tâche parallèle à celle contenant l’instruction TRIGGER.

9-10-136- TRIGGERS – Active le mouvement déclenché

Cette instruction déclenche le mouvement déclenché sans aucune condition.

Cette instruction doit être utilisée dans une tâche parallèle à celle contenant l’instruction TRIGGER.

9-10-137- VEL - Vitesse

Syntaxe : VEL = <Expression>

Unité : Expression : unité utilisateur par seconde (Ex : mm/s, tr/s, degré/s).

Types acceptés :Expression : réel

Description : Cette valeur spécifie la vitesse courante en unité par seconde.

Remarques : <Expression> doit être une expression réelle valide. Cette valeur de vitesse peut être modifiée à tout moment.

Exemple : VEL = 2000

Voir aussi : ACC, DEC, POS

9-10-138- VEL_S

Syntaxe : VEL_S

Description : Cette fonction retourne la vitesse courante.

Exemple : STTA = 100

IF VEL_S<50 GOTO ARRET

Voir aussi : POS_S

9-10-139- VEL%

Syntaxe : VEL% = <Expression>

Limite : Expression : de 0 à 100

Types acceptés :Expression : octet

Description : Cette fonction ajuste la vitesse courante en pourcentage du paramètre de vitesse de l'écran Motion control / Configuration / Profil de vitesse.

Exemple : VB0 = 50

VEL% = VB0

Voir aussi : ACC%, DEC%

9-10-140- VELMASTER_S

Syntaxe : VELMASTER_S

Description : Cette fonction retourne la vitesse courante de l'axe maître.

Exemple : GEARBOX(1,1)

IF VELMASTER_S<50 GOTO ARRET

Voir aussi : VEL_S

9-10-141- VERSION – Version de l'operating system (Firmware)

Syntaxe : <VI n°XX>=VERSION

Description : Cette fonction retourne la version de l'Operating System.

9-10-142- VIRTUALMASTER – Active/désactive le maître virtuel

Syntaxe : VIRTUALMASTER ON/OFF

Description : Cette instruction permet de déclarer l'axe maître en mode virtuel. Toutes les instructions de positionnement MOVA, MOVR, STTA, SSTR seront « exécutées par le maître », l'axe maître se « déplacera » virtuellement. Il est possible de réaliser des fonctions de synchronisations entre le maître et l'esclave (le moteur) en utilisant les instructions MOVS, GEARBOX ….

Attention : Pour utiliser cette instruction, sélectionner « virtuel » comme source du maître dans la fenêtre Motion control \ Maître/esclave.

9-10-143- WAIT - Attente d'une condition

Syntaxe : WAIT <Condition>

Description : Cette instruction permet au système d'attendre que la condition soit vraie.

Exemple : WAIT INP(11)=On 'Attente passive

9-10-144- WRITECAM – Permet d'écrire un point de came

Syntaxe : WriteCam(<Index>, <Sous index>)=<VRx>

Description : Cette fonction permet d'écrire un point de came dans la mémoire FRAM

Limites : <Index> de 0 à 511, numéro du point de came en FRAM

<Sous index> de 0 à 3, paramètre du point de came :

? 0 pour la position de maître

? 1 pour la position de l'esclave

? 2 pour la tangente maître

? 3 pour la tangente esclave

<VRx> de VR0 à VR255

Voir aussi : WRITECAM

9-10-145- WRITEI - Ecriture d'un entier en FRAM

Ecriture : WRITEI (<Adresse>) = <VIn ou valeur>

Limites : <Adresse> : de 0 à 4095

n de 0 à 255

9-10-146- WRITEL - Ecriture d'un entier long en FRAM

Ecriture : WRITEL (<Adresse>) = <VLn ou valeur>

Limites : <Adresse> : de 0 à 4094

n de 0 à 255

Attention : La lecture et l'écriture d'un entier long utilise 2 adresses mémoires consécutives (adresse et adresse+1).

9-10-147- WRITEPARAM – Ecriture d'un paramètre

Syntaxe : READPARAM (<Index>, <Sub-Index>) = <Variable>

Types acceptés :<Variable> du type entier long

<Index> de type entier

<Sub-Index> de type octet

Description : Cette fonction permet de lire via le bus CANopen, les paramètres du variateur.

Exemple : WRITEPARAM(9984,6) = 1 'Active le modulo sur l'axe

9-10-148- WRITER - Ecriture d'un réel en FRAM

Ecriture : WRITER (<Adresse>) = <VRn ou valeur>

Limites : <Adresse> : de 0 à 4094

n de 0 à 255

Attention : La lecture et l'écriture d'un réel utilise 2 adresses mémoires consécutives (adresse et adresse+1).

9-10-149- XOR – Opérateur ou exclusif

Syntaxe : <Expression1> XOR <Expression2>

Types acceptés :Expression1, Expression2 : Bit, Octet, Entier

Description : Cette fonction fait un Ou Exclusif entre les expressions.

Remarques : <Expression1> et <Expression2> doivent être du même type de donnée. Cette fonction restitue le type de donnée de <Expression1>.

Exemple : IF VL1 XOR 0FF00h ...

Voir aussi : AND, OR, NOT, IF

10- Annexes

10-1- Afficheur STATUS 7 segments

10-1-1- Description des messages :

A) A la mise sous tension du variateur :

1. **Phase d'initialisation du BOOT :**

Avant d'initialiser le BOOT, l'afficheur affiche :

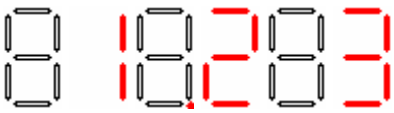
En cas de défaut d'initialisation, on pourra avoir les défauts suivants :

- : erreur de checksum du secteur de boot ou de l'OS, le secteur boot et/ou l'OS ont été altérés.
- : l'OS n'est pas correctement chargé
- : erreur interne

2. **Phases d'initialisation du système d'exploitation :**

Les segments s'allument très rapidement dans l'ordre suivant :

En fin d'initialisation du système d'exploitation, on affiche le numéro de version :



L'exemple ci dessus donne le numéro de version 1.23

3. Après les initialisions :

La sortie variateur prêt s'active (S1), si le iDPL est utilisé : les tâches automatiques sont lancées et il ne doit rester qu'un point qui clignote.

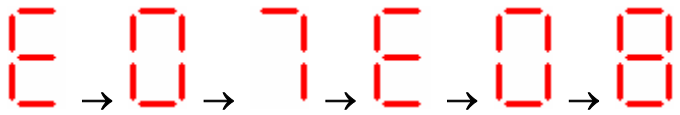
- Si le iDPL n'est pas utilisé, on fait tourner les segments en même temps et dans le même sens que le moteur.
- Si le iDPL est utilisé, on laisse uniquement le point . Les segments sont modifiés par l'exécution de l'instruction Display dans le programme iDPL.

B) Pendant l'utilisation du variateur :

1. Sur apparition d'une erreur :

Les numéro des erreurs détectés sont affichés par ordre croissant puis rebouclage sur la première erreur.

Ex : Pour une erreur de température moteur E7 et une erreur de codeur E8, on aura

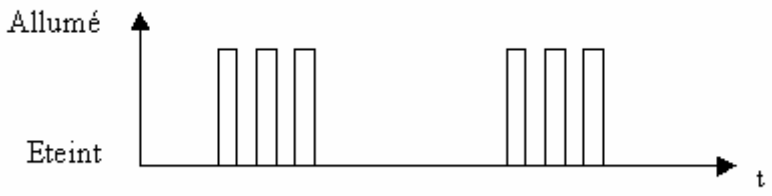


2. Sur disparition d'un défaut :

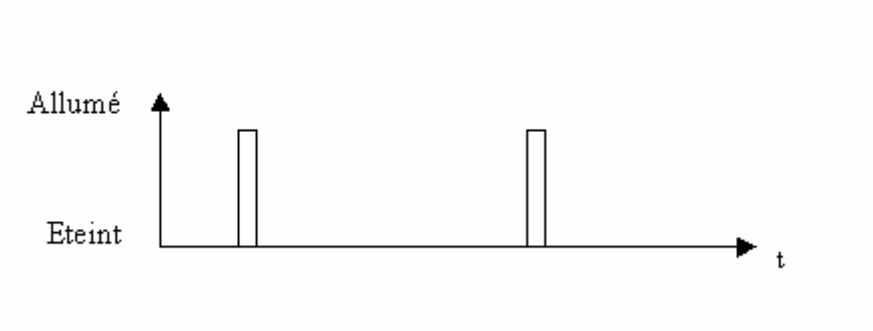
Disparition du numéro d'erreur et retour à un affichage normal (comme après l'initialisation)

Clignotement du point :

- Si liaison système en cours (signal RTS monté) :



- Si pas de liaison système en cours :

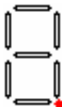


C) Pendant un rechargement du system d'exploitation :

Erase : effacement de la FLASH

Flash : écriture de la flash

Read : lecture de la flash

 Reboot

D) Lors d'un opération sur la FLASH :

est affiché sur l'afficheur pendant le durée du flashage (lors d'une sauvegarde de paramètres, d'un programme ou des variables).

10-1-2- Messages d'erreur :

A) Liste des erreurs :

Sur-tension DCBus : une sur-tension a été détectée sur le bus continu interne. Ce défaut peut être dû à une sur-tension sur le réseau ou à la résistance de ballast qui n'est pas suffisante

Sous-tension DCBus : une tension minimale a été détectée sur le bus continu interne. Ce défaut est géré pendant que le variateur est activé 'ENABLE = ON, tension DC Bus inférieur à un paramètre) et lors de la demande d'asservissement (tension DC Bus inférieur à 250V)

I²t moteur : I²t moteur détecté.

Sur-courant : un courant supérieur au courant maximal mesurable a été détecté.

Court-circuit : un court-circuit entre phases ou la mise à la terre d'une phase du moteur a été détecté.

Température IGBT : température maximale atteinte dans le variateur.

Température moteur : température maximale atteinte dans le moteur.

Erreur résolveur : Signaux résolveur ou codeur absolu ou SINCOS défectueux.

Paramètres invalides : erreur de checksum sur les paramètres du variateur ou paramètres non initialisés.

Défaut modèle de variateur : le fichier de paramètre ne correspond pas au modèle de variateur ou paramètres non configurés.

Erreur iDPL : une erreur a été détecté pendant l'exécution des tâches iDPL (division par zéro, instruction incorrect, problème de CAM ou de mouvement synchro …).

Erreur de poursuite : le variateur a dépassé l'erreur de poursuite.

Erreur Mémoire Flash : écriture impossible. Contacter notre service technique.

Erreur FPGA : chargement impossible. Contacter notre service technique.

Survitesse ou erreur CAN : le moteur a dépassé la vitesse nominale du moteur en mode couple ou le temps de réponse du contrôleur CAN est dépassé.

Erreur saturation résolveur. Signaux résolveur ou SINCOS trop élevés.

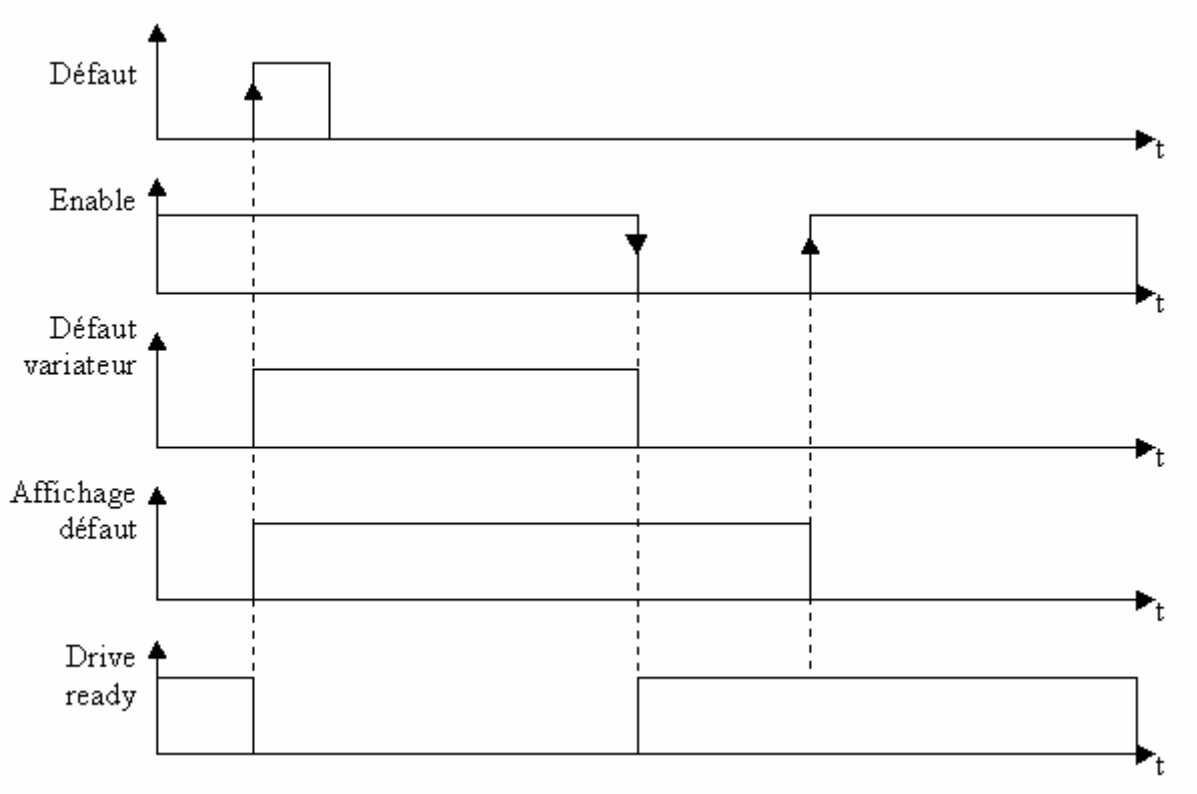
Erreur alimentation 24V. Ce défaut ce déclenche si l'alimentation est mal ou pas filtré. Vérifier l'alimentation 24V.

B) Liste des erreurs iDPL:

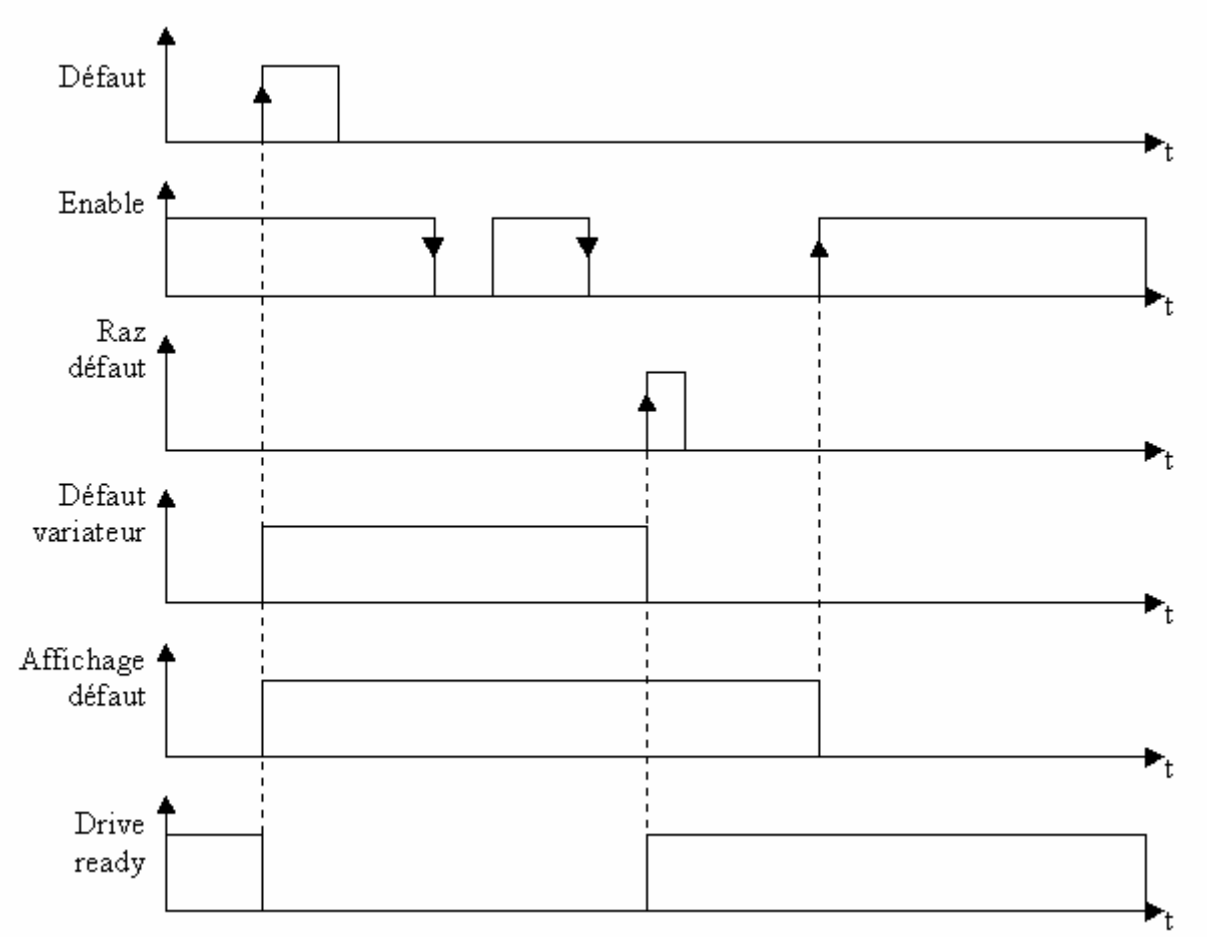
Erreur 1 :	Instruction illégal. Corruption de la flash et/ou erreur de compilation.
Erreur 2 et 3 :	Erreur allocation came. Erreur interne à l'OS.
Erreur 4 :	Point de came impossible à calculer. Réduire la longueur du maître en fractionnant la came par exemple.
Erreur 5 :	Fonction illégal. Corruption de la flash et/ou erreur de compilation.
Erreur 6 :	Division par 0. Une division par 0 a été faite au niveau du iDPL.
Erreur 7 :	Erreur sur le numéro de came dans une instruction LOADCAM
Erreur 8 :	Erreur sur un offset en FRAM (valeur en dehors de 0 et 4095)
Erreur 9 :	Erreur sur un point de came (distance maître ou esclave négative)
Erreur 10 :	Numéro de tâche invalide

C) Suppression des défauts :

- Si l'entrée E4 n'est pas utilisée en RAZ défaut , procéder comme suivant :



- Si l'entrée E4 est utilisée en RAZ défaut , procéder comme suivant :



10-2- CANopen

10-2-1- Définition

A) Introduction

Le bus CAN (Controller Area Network) est apparu au milieu des années 80 pour répondre aux besoins de la transmission de données dans le secteur automobile. Ce type de bus permet d'obtenir des taux de transfert élevés.

Les spécifications du CAN définissent 3 couches parmi le modèle OSI : la couche physique, la couche liaison des données et la couche application. La couche physique définit le mode de transmission des données en fonction du support de transmission. La couche liaisons des données représente le noyau du protocole CAN puisque cette couche est responsable de la trame à envoyer, de l'arbitrage, de la détection des erreurs, etc… . La dernière couche est la couche application appelée aussi CAL (CAN Application Layer). Celle-ci est donc une description générale du langage pour les réseaux CAN qui offre de nombreux services de communication.

CANopen est un type de réseau qui est basé sur le système du bus série et de la couche application CAL. CANopen ne propose qu'une partie des services de communication offerte par CAL. Ce sont les avantages nécessaires dont ont besoin les ordinateurs ayant des performances réduites et des capacités de stockage faible.

Le CANopen est, par conséquent, une couche application standardisée par les spécifications du CIA (CAN In Automation) : DS-201…DS-207.

Le gestionnaire du réseau permet une initialisation simplifiée du réseau. Le réseau peut être étendu avec tous les composants que l'utilisateur désire.

Le bus CAN est un bus multi-maître. Contrairement aux autre bus de terrain, ce sont les messages qui sont identifiés et non les modules connectés. Les éléments du réseau sont autorisés à envoyer leurs messages à chaque fois que le bus est libre. Les conflits sur le bus sont résolus par un niveau de priorité donné aux messages. Le bus CAN émet des messages qui sont divisés en 2032 niveaux de priorités. Tous les éléments du réseau ont les mêmes droits et donc cette communication n'est seulement possible que sans bus maître.

Chaque élément décide lui-même lorsqu'il veut envoyer des données. Il est cependant possible de faire envoyer des données par un autre élément. Cette demande est effectuée par la trame distante.

Les spécifications du CANopen (DS-201…DS-207) définissent les caractéristiques techniques et fonctionnelles que nécessitent un appareil individuel pour être associé sur le réseau. Le bus CANopen fait une distinction entre les appareils serveurs et les appareils clients.

B) La communication CANopen

Le profil de la communication du CANopen permet de spécifier les informations pour l'échange de données en temps réel et des paramètres. Le CANopen utilise des services optimisés suivant les différentes sortes de données.

↳ PDO (Process Data Object)

⇨Echange de données en temps réel

⇨Identifiant à haute priorité

⇨ Transmission synchrone ou asynchrone

⇨ Maximum de 8 octets (un message)

⇨ Format prédéfini

↳ SDO (Service Data Object)

⇨ Accède au dictionnaire des objets d'un appareil

⇨ Identifiant à basse priorité

⇨ Transmission asynchrone

⇨ Données distribuées dans plusieurs télégrammes

⇨ Données adressées par un index

Les caractéristiques diffusées par le CAN sont reçues et évalués par tous les appareils connectés. Chaque service d'un appareil CAN est paramétré par un COBID (Communication OBject Identifier). Le COBID est un identifiant qui caractérise le message. C'est ce paramètre qui permet d'indiquer à un appareil si le message doit être traiter. Pour chaque service (PDO ou SDO), il est nécessaire de spécifier un COBID à l'émission (envoi d'un message) et un COBID à la réception (récupération de message). Pour le premier SDO serveur, le COBID est fixe est ne peut pas être modifié à distance. De plus, il est calculé à partir du NODE-ID. Le NODE-ID est le paramètre qui caractérise l'appareil et qui permet d'accéder de façon unique à l'appareil.

PDO (Process Data Object)

C'est un échange de données arbitré entre deux modules. Les PDO peuvent transférer alternativement des synchronisations ou des événements contrôlés pour réaliser la demande d'envoi des messages. Avec le mode d'événements contrôlés, la charge du bus peut être réduite au minimum. Un appareil peut, donc, réaliser une communication à haute performance avec un faible taux de transfert.

L'échange de donnée avec le PDO utilise les avantages du CAN :

↳ L'envoi de message peut être déclenché par un événement asynchrone. (événements contrôlés)

↳ L'envoi de message peut être déclenché sur la réception d'un événement de synchronisation.

↳ Récupération par une trame à distance.

SDO (Service Data Object)

C'est un échange de données point à point. Un appareil vient faire une demande d'accès dans la liste d'objets d'un SDO. Le SDO renvoie une information correspondant au type de requête fait par le demandeur. Chaque SDO peut être client et/ou serveur. Un SDO serveur ne peut pas faire de demande envers un autre SDO par contre lui peut répondre à toute demande d'un SDO client. Contrairement aux PDO, les SDO doivent suivre un protocole de communication particulier. La trame envoyée est composée de 8 octets :

↳ Domain Protocol (Octet 0) : il définit la commande (Upload, Download,….).

- Index sur 16 bits (Octet 1 et 2) : il définit l'adresse du dictionnaire des objets.
- Sub-index sur 8 bits (Octet 3) : il définit l'élément de l'objet sélectionné dans le dictionnaire.
- Paramètre (Octet 4 à 7) : Il définit la valeur du paramètre lu ou écrit.

Le gestionnaire de réseau comporte un mode simplifié de démarrage du réseau. La configuration du réseau n'est pas nécessaire dans tous les cas. La configuration par défaut des paramètres est donc parfois suffisante. Si l'utilisateur désire optimiser le réseau CANOpen ou augmenter ses fonctionnalités, il peut alors modifier lui-même ces paramètres. Dans les réseaux CANOpen, tous les appareils ont les mêmes droits et l'échange des données est directement régulé entre chaques appareils participants.

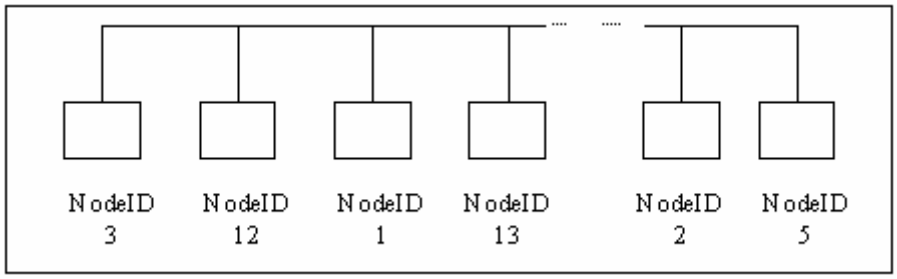
Le profil d'un appareil définit les paramètres nécessaires pour une communication. Le contenu de ce profil est spécifié par le constructeur. Les appareils ayant le même profil sont directement interchangeables. La plupart des paramètres sont décrits par le constructeur. Le profil possède aussi des emplacements vides qui correspondent aux futures extensions de fonctionnalités des constructeurs.

Dans la plupart des bus maître/esclave, l'efficacité du maître détermine le comportement de tout le réseau. De plus, les esclaves ne peuvent pas directement communiquer entre eux. Toutes ces caractéristiques augmentent, donc, le nombre d'erreurs de transmission. CANOpen élimine tous ces désavantages. Le comportement temporel peut être spécifié individuellement pour chaque tâche respective des appareils participants. Ainsi, le système entier de communication n'a pas besoin de plus d'efficacité si seulement certains appareils participants nécessitent plus de performance. De plus, une tâche automatique peut être séparée pour chacun des appareils participants. Ainsi, les performances disponibles du contrôleur du réseau peuvent être utilisées de manière optimales et peuvent être augmentées à tout instant par adjonction de nouveaux appareils participants.

Le mapping des variables utilisées lors des échanges de type PDO permet d'utiliser de manière optimale la bande passante actuelle du bus. CANopen détermine les valeurs en défaut de tous les paramètres.

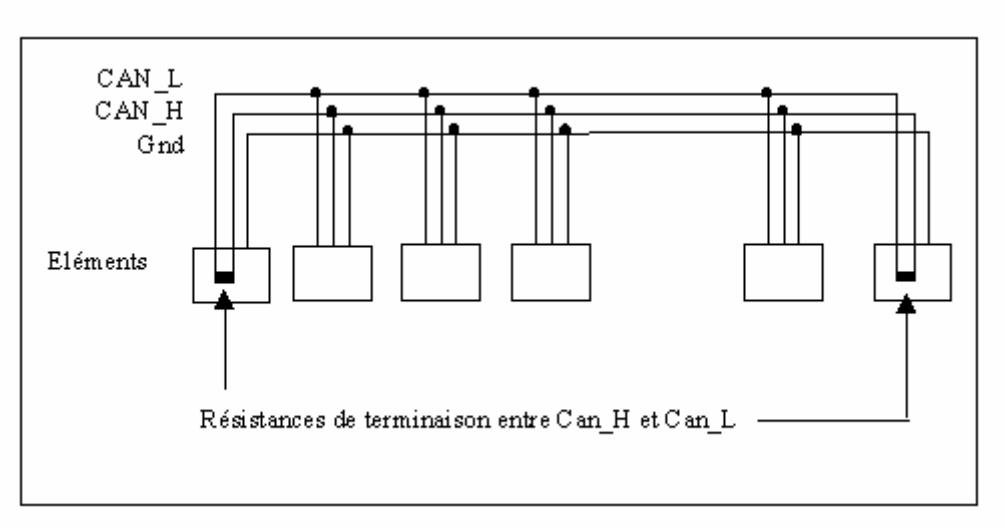
C) Configuration du réseaux

Le réseau CAN OPEN est constitué de différents éléments, tous pouvant être maître et esclaves. Ils sont identifiés dans le réseau par un numéro arbitraire, que l'on appelle le Node-ID. Ce paramètre doit être unique : 2 éléments distincts du réseau ne peuvent pas avoir le même Node-ID. Ce Node-ID est très important, c'est la véritable carte d'identité du périphérique sur le réseau CAN Open.



Exemple de configuration de réseau Can Open

Le principe de câblage est le suivant :



Principe de câblage d'un réseau Can Open

Important : Pensez aux résistances de terminaison à chaque extrémité du réseau

D) Type de messages envoyés

Il y a deux grandes familles de messages envoyés par la liaison Can Open :

- Les SDO (Service Data Object) transportent des données

- Les PDO (Process Data Object) transmettent des événements.

10-2-2- Carte IMDCANI pour drive IMD

A) Présentation de la carte IMDCANI

Les divers paramètres du drive IMD SCAN ainsi que les variables sont incluses dans un élément à 2 dimensions, **le dictionnaire**.

Chaque paramètre ou paramètre est défini par une adresse d'index et une adresse de SubIndex.

Le drive IMD peut communiquer avec un autre élément du réseau de plusieurs façons. Il peut mettre à disposition des données en les écrivants dans sa table locale : n'importe quel élément du réseau peut alors lire et même écrire sur cette table locale. C'est la méthode employée par exemple pour communiquer avec un pupitre intelligent de type Dialog 80 ou Dialog 640.

Le drive IMD peut également lire ou écrire une table locale d'un autre élément. Cette opération se réalise alors par les instructions SDOx et Vx.

B) Caractéristiques

- Un serveur SDO par défaut pour le paramétrage de la carte à distance par un superviseur.
- Un client SDO pour accéder aux variables et aux paramètres des périphériques CANopen tels que des pupitres, automates et cartes PC.
- 8 PDO en émission pour piloter les sorties des modules I/O ou signaler un événement à une autre IMD.
- 8 PDO en réception pour recevoir les entrées des modules I/O ou recevoir les événements à une autre IMD.
- Des fonctions d'accès direct au bus CAN pour envoyer et recevoir des messages spécifiques tels que les fonctions NMT et DBT.
- Des fonctions de Node Guarding.

C) Raccordement

a) Affectation et brochage :

X2 et X3: Extension: Bus de communication optionnel RJ45

N°	Module CANopen X2	Module CANopen X3
1		
2		
3		
4		
5	GND	GND
6		
7	CAN_L	CAN_L
8	CAN_H	CAN_H
	SHIELD - Raccorder la tresse blindée sur le corps du SUBD	

- Les deux connecteurs X2 et X3 sont identiques et contiennent les mêmes signaux. Ils facilitent la mise en réseau de plusieurs variateurs

- Numéro d'adresse (NodeID): le NodeID correspond à la valeur des 5 premiers dipswitchs + 1

Ex: dipswitchs: 1 -> ON, 2 -> OFF, 3 -> ON, 4 -> OFF, 5 -> OFF

Valeur dipswitchs = 1 + 4 = 5

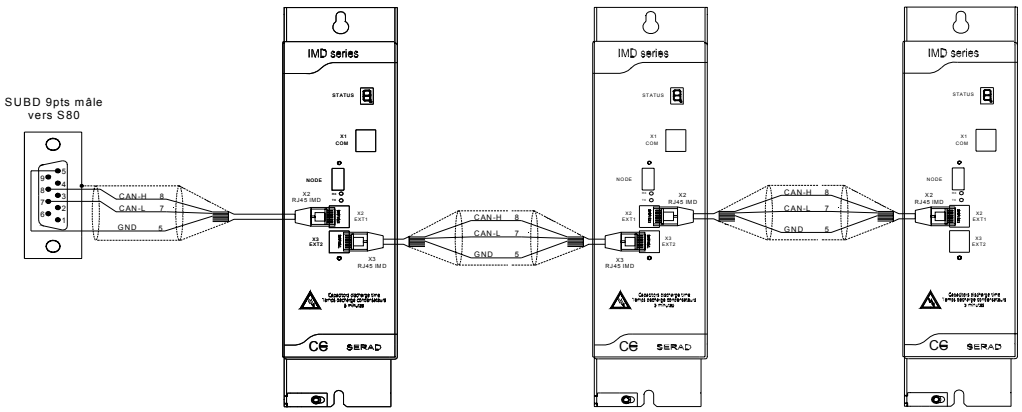
NodeID = 5 + 1 = 6

- La validation des résistances de terminaison du bus (120Ω) se fait en activant le dipswitch 6 sur la position ON.

b) Vitesses maximales de transmission en fonction de la longueur du réseau CAN Open :

Vitesse maximale de transmission	Longueur du bus
10K à 125 kBauds	500 m
250 kBauds	250 m
500 kBauds	100 m
800 kBauds	50 m
1 Mbauds	25 m

c) Exemple avec 3 drive IMD et 1 SUPERVISOR :



D) Diagnostic du bus

LED CAN Rx/Tx:

Elles clignotent proportionnellement au débit sur le bus CAN (son intensité peut donc être très faible ou très forte)

E) Dictionnaire du CANOpen :

- Le variateur gère le mode SDO et PDO pour accéder en lecture / écriture à ses paramètres et ses variables mais aussi à celle d'un périphérique CANOpen.

Index	Sub-idx	Nom	Type	Attr.	Défaut	Description
1000	0	Device type	32 bits non signé	ro	403	type d'appareil
1001	0	Error register	32 bits non signé	ro	0	registre d'erreur interne
1002	0	Manufacturer Status Register	32 bits non signé	ro	0	registre d'etat spécifique au constructeur
1003	0	predefined error field	8 bits non signé	ro	1	nombre d'erreurs apparues
	1	actual error	32 bits non signé	ro	0	dernière erreur apparue

1004	0	number of PDO's supported	32 bits non signé	ro	00080008h	Nombre de PDO supporté
	1	Number of synchronous PDO	32 bits non signé	ro	0	Nombre de PDO synchrone supporté
	2	Number of asynchronous PDO	32 bits non signé	ro	00080008h	Nombre de PDO asynchrone supporté
1005	0	COB-ID	32 bits non signé	rw	00000008h	COB-OD SYNC message
100B	0	Node ID	32 bits non signé	ro	aucune	N° de noeud local
100C	0	Guard time	16 bits non signé	rw	aucune	durée en ms
100D	0	Life time factor	8 bits non signé	rw	aucune	Timeout = Guard time x Life time factor
100E	0	Node guarding ID	32 bits non signé	rw	700h + NodeID	COB-ID Nodeguarding
100F	0	Number of SDO's supported	32 bits non signé	ro	00010001h	Nombre de SDO supporté
1200	0	Number of elements	8 bits non signé	ro	2	paramètre du 1er SDO serveur
	1	SDO receive COB-Id	32 bits non signé	ro	600h+node-ID	COB-ID de récéption du 1er SDO serveur
	2	SDO transmit COB-ID	32 bits non signé	ro	580h+node-ID	COB-ID d'envoi du 1er SDO serveur
	3	node ID of the SDO client	8 bits non signé	rw	none	Node ID du SDO client
1280	0	Number of elements	8 bits non signé	ro	2	paramètre du 1er SDO client
	1	SDO receive COB-Id	32 bits non signé	ro	aucune	COB-ID de récéption du 1er SDO client
	2	SDO transmit COB-ID	32 bits non signé	ro	aucune	COB-ID d'envoi du 1er SDO client
	3	node ID of the SDO server	8 bits non signé	rw	none	Node ID du SDO serveur
1400	0	Number of elements	8 bits non signé	ro	2	paramètre de réception du 1er PDO
	1	COB-ID	32 bits non signé	rw	aucune	COB-ID utilisé par le PDO
	2	Transmission type	8 bits non signé	rw	254	Type de la réception
…						
1407						paramètre de réception du 8ème PDO
1800	0	Number of elements	8 bits non signé	ro	2	paramètre d'émission du 1er PDO
	1	COB-ID	32 bits non signé	rw	aucune	COB-ID utilisé par le PDO

	2	Transmission type	8 bits non signé	rw	254	Type de l'émission 252->sur synchro 253->remote(RTR) 254->périodique 255->sur modification
	3	Inhinit time	16 bits non signé	rw	254	durée d'inhibition (ms)
…						
1807						paramètre d'émission du 8ème PDO

- Le dictionnaire contient les différents paramètres et variables du variateur.

Voir l'écran Aide \ Modbus-CANopen

10-2-3- Liste des instructions

A) Liste des instructions CANopen

Instructions d'échange de variables entre drive

VF	Lecture ou écriture d'une variable distante (bit)
VB	Lecture ou écriture d'une variable distante (octet)
VI	Lecture ou écriture d'une variable distante (word)
VL	Lecture ou écriture d'une variable distante (entier long)
VR	Lecture ou écriture d'une variable distante (réel)

Lecture et écriture du dictionnaire

CANOPENB	Lecture ou écriture d'un paramètre (octet)
CANOPENI	Lecture ou écriture d'un paramètre (word)
CANOPENL	Lecture ou écriture d'un paramètre (entier long)

Instructions en mode SDO

SDOB	Lecture ou écriture d'une variable distante (octet)
SDOI	Lecture ou écriture d'une variable distante (word)
SDOL	Lecture ou écriture d'une variable distante (entier long)
SDOBX	Lecture ou écriture d'une variable distante (octet) d'un périphérique
SDOIX	Lecture ou écriture d'une variable distante (word) d'un périphérique
SDOLX	Lecture ou écriture d'une variable distante (entier long) d'un périphérique

Instructions en mode PDO

CANSENDNMT	Envoie un NMT sur le bus CAN
CANSENDSYNCHRO	Envoie 1 message de synchronisation
CANSETUPSYNCHRO	Initialise la synchronisation
PDOEVENT	Test l'arrivée d'un PDO
PDOTX	Envoie des éléments mappés

Instructions en mode CAN générique

CAN	Lecture ou écriture des données
CANERR	Détection des erreurs
CANERRCOUNT	Contrôle et efface les erreurs de la communication
CANEVENT	Test de l'arrivée d'un message
CANTX	Envoie d'un message
SETUPCAN	Paramétrage d'un message

Instructions pour le multiaxe

CANPOSSTATUS	Retourne l'état de la réception de la position par CAN
CANPOSTIMEOUTRAZ	Acquitte le défaut TIMEOUT de CANPOSSTATUS
STARTCANRECEIVEPOSITION	Démarre la réception de la position d'un axe par bus CAN
STARTCANSENDPOSITION	Démarre l'envoie de la position de l'axe sur le bus CAN
STOPCANRECEIVEPOSITION	Arrête la réception de la position d'un axe par bus CAN
STOPCANSENDPOSITION	Arrête l'envoie de la position de l'axe sur le bus CAN

B) CAN – Lecture et écriture d'un message

Syntaxe 1 : CAN (<NuméroOctet>) = <Variable>

Syntaxe 2 : <Variable> = CAN (<NuméroOctet>)

Types acceptés :<Variable> : chaîne de caractères

<NuméroOctet>

Description : Cette fonction permet de lire ou d'envoyer un message.

Remarques : Il est nécessaire de paramétrer le COBID de réception pour pouvoir recevoir un message.

C) CANERRCOUNTER - Contrôle et efface les erreurs de la communication

Syntaxe 1 : <Variable> = CANERRCOUNTER

Syntaxe 2 : CANERRORCOUNTER = 0

Limites : <Variable> : de 0000h à FFFFh

Types acceptés :<Variable> : entier

Description : La syntaxe 1 permet de connaître le nombre d'erreur qui se sont produites depuis la dernière initialisation du compteur. La deuxième syntaxe permet d'initialiser le compteur d'erreur.

D) CANERR – Détection des erreurs

Syntaxe 1: <Variable> = CANERROR

Syntaxe 2: CANERROR = 0

Types acceptés :<Variable> : octet

Bit 0 à 1 si erreur du bus

Bit 1 à 1 si le temps de réponse du SDO est écoulé

Bit 2 à 1 si erreur de Node Guarding

Description : Cette fonction permet de détecter si une erreur s'est produite sur le bus CAN.

E) CANEVENT – Test l'arrivée d'un message

Syntaxe : <Variable> = CANEVENT

Types acceptés : <Variable> : booléen

Description : Cette fonction permet de savoir si un message a été réceptionné.

Remarques : Il est nécessaire de paramétrer le COBID de réception pour pouvoir recevoir un message.

F) CANOPENX - Lecture ou écriture d'un paramètre

Syntaxe 1 : CANOPENB (<Index>, <Sub-Index>) = <octet ou variable>

Syntaxe 2 : <Variable> = CANOPENB (<Index>, <Sub-Index>)

Syntaxe 3 : CANOPENI (<Index>, <Sub-Index>) = <entier ou variable>

Syntaxe 4 : <Variable> = CANOPENI (<Index>, <Sub-Index>)

Syntaxe 5 : CANOPENL (<Index>, <Sub-Index>) = <entier long ou variable>

Syntaxe 6 : <Variable> = CANOPENL (<Index>, <Sub-Index>)

Limites : <Index> : de 0000h à FFFFh

<Sub-index> : de 00h à FFh

Syntaxe 1 et 2 : <Variable> : de 00h à FFh

Syntaxe 3 et 4 : <Variable> : de 0000h à FFFFh

Syntaxe 5 et 6 : <Variable> +/- 7FFFFFFFh

Description : Cette fonction permet de lire ou d'écrire une donnée dans le dictionnaire du variateur IMD.

G) CANPOSSTATUS - Retourne l'état de la réception de la position par CAN

Syntaxe : CANPOSSTATUS

Description : Cette instruction retourne l'état de la réception de la position par CAN

- 0 : pas de réception en cours
- 1 : réception en cours
- 2 : la réception a été interrompue pendant une durée supérieure à <TimeOut> mais à repris.
- 3 : la réception est stoppée suite à un décalage trop important détecté par rapport au maître.

H) CANPOSTIMEOUTRAZ - Acquitte le défaut TIMEOUT de CANPOSSTATUS

Syntaxe : CANPOSTIMEOUTRAZ

Description : Cette instruction permet d'acquitter le défaut <TimeOut> de CANPOSSTATUS (si CANPOSSTATUS retournait 2, il repasse à 1)

I) CANSENDNMT – envoie un NMT sur le bus CAN

Syntaxe : CANSENDNMT (<Noeud>, <Action>)

Description : Cette instruction permet d'envoyer la commande NMT aux périphériques spécifiés par le numéro de <Noeud> pour démarrer les PDO.

Valeurs acceptées : <Noeud> 0 à 31

- 0 : envoie à tous les périphériques ainsi que lui même
- n° drive local : envoie un NMT à lui même
- autre : envoie un NMT aux périphériques spécifiés

<Action>

- 1 : envoie un START
- 2 : envoie un STOP
- 3 : envoie un DTSCONNECT

J) CANSENDSYNCHRO – Envoie 1 message de synchronisation

Syntaxe : CANSENDSYNCHRO (<COBID>)

Description : Cette instruction permet d'envoyer 1 message de synchro.

Valeurs acceptées : <COBID> entre 0x80 et 0xFF (0x80 par défaut)

K) CANSETUPSYNCHRO – Initialise la synchronisation des messages PDO

Syntaxe : CANSETUPSYNCHRO (<COBID>, <Période>)

Description : Cette instruction permet d'initialiser la synchro des messages PDO.

Valeurs acceptées : <COBID> entre 0x80 et 0xFF (0x80 par défaut)

< Période> nombre de période de 150µs entre 2 même PDO.

Attention : Si <Période> = 0 alors la synchronisation est arrêtée.

L) CANTX - Envoie d'un message

Syntaxe : CANTX

Description : Cette fonction envoie le message initialisé par la fonction CAN.

M) PDOEVENT – Test l'arrivée d'un PDO

Syntaxe : <Variable> = PDOEVENT (<N°PDO>)

Limites : <N°PDO> : de 01h à 08h

Types acceptés :<Variable>, <N°PDO> : Octet

Description : Cette fonction permet de connaître si une demande d'un PDO est effective.

Remarques : Il est nécessaire de préciser les paramètres de transmission du PDO pour pouvoir recevoir un PDO.

N) PDOTX – Provoque l'envoie des éléments mappés

Syntaxe : PDOTX

Description : Cette fonction envoie les éléments PDO mappés.

O) SDOB, SDOI et SDOL - Lecture ou écriture d'une variable distante

Syntaxe 1 : SDOB (<Index>, <Sub-Index>) = <octet ou variable>

Syntaxe 2 : <Variable> = SDOB (<Index>, <Sub-Index>)

Syntaxe 3 : SDOI (<Index>, <Sub-Index>) = <entier ou variable>

Syntaxe 4 : <Variable> = SDOI (<Index>, <Sub-Index>)

Syntaxe 5 : SDOL (<Index>, <Sub-Index>) = <entier long ou variable>

Syntaxe 6 : <Variable> = SDOL (<Index>, <Sub-Index>)

Limites : <Index> : de 0000h à FFFFh

<Sub-index> : de 00h à FFh

Syntaxe 1 et 2 : <Variable> : de 00h à FFh

Syntaxe 3 et 4 : <Variable> : de 0000h à FFFFh

Syntaxe 5 et 6 : <Variable> +/- 7FFFFFFFh

Description : Cette fonction permet de lire ou d'écrire une variable à distance dans le dictionnaire du variateur IMD.

P) SDOBX, SDOIX et SDOLX - Lecture ou écriture d'une variable distante

Syntaxe 1 : SDOBX (<Index>, <Sub-Index>, <Adresse>) = <octet ou variable>

Syntaxe 2 : <Variable> = SDOBX (<Index>, <Sub-Index>, < Adresse >)

Syntaxe 3 : SDOIX (<Index>, <Sub-Index>, < Adresse >) = <entier ou variable>

Syntaxe 4 : <Variable> = SDOIX (<Index>, <Sub-Index>, < Adresse >)

Syntaxe 5 : SDOLX (<Index>, <Sub-Index>, < Adresse >) = <entier long ou variable>

Syntaxe 6 : <Variable> = SDOLX (<Index>, <Sub-Index>, < Adresse >)

Limites : <Index> : de 0000h à FFFFh

<Sub-index> : de 00h à FFh

Syntaxe 1 et 2 : <Variable> : de 00h à FFh

Syntaxe 3 et 4 : <Variable> : de 0000h à FFFFh

Syntaxe 5 et 6 : <Variable> +/- 7FFFFFFFh

Description : Cette fonction permet de lire ou d'écrire une variable à distance dans le dictionnaire du variateur IMD et l'envoyer vers un périphérique précis.

Equivalent à un SETUPCAN suivi d'un SDOx

Q) SETUPCAN - Paramétrage d'un message

Syntaxe : SETUPCAN (<TX COBID>, <RX COBID>)

Types acceptés :<TX COBID>, <RX COBID> : entier long

Description : Cette fonction permet de configurer les COBID de réception et de transmission avant l'envoi d'un message.

R) STARTCANRECEIVEPOSITION - Démarre la réception de la position d'un axe par bus CAN

Syntaxe : STARTCANRECEIVEPOSITION (<PDO>, <COBID>, <Offset>, <TimeOut>)

Description : Cette instruction démarre la réception de la position d'un axe par bus CAN.

Valeurs acceptées : < PDO> numéro de PDO de 1 à 8

<COBID> entre 0x181 et 0x37F

<Offset> permet de compenser le délai de transmission, entre 0 et la période d'envoi de la position sur le bus CAN

- <Offset> = 0 : très bonne précision mais avec un décalage temporel égale à la période d'émission de la position
- <Offset> = <Période> + 1: décalage temporel très faible ou nul mais précision moins bonne dû à l'interpolation.

<TimeOut> nombre de période de 150µs avant que CANPOSSTATUS passe en défaut.

Attention : Le PDO utilisé par cette instruction ne peut pas être utilisé par une autre instruction CAN.

L'utilisation des instructions SAVEPARAM et SAVEVARIABLE provoque l'arrêt de la réception de la position.

S) STARTCANSENDPOSITION - Démarre l'envoie de la position de l'axe sur le bus CAN

Syntaxe : STARTCANSENDPOSITION (<Source>, <PDO>, <COBID>, <Période>)

Description : Cette instruction démarre l'envoi de la position de la <Source> sur le bus CAN.

Valeurs acceptées : <Source> 0 pour l'axe esclave et 1 pour l'axe maître

< PDO> numéro de PDO de 1 à 8

<COBID> entre 0x181 et 0x37F

<Période> nombre de période de 150µs entre 2 même PDO.

Attention : Si <Période> = 0 alors la position sera envoyé le plus souvent possible.

Le PDO utilisé par cette instruction ne peut pas être utilisé par une autre instruction CAN.

L'utilisation des instructions SAVEPARAM et SAVEVARIABLE provoque l'arrêt de la réception de la position.

T) STOPCANRECEIVEPOSITION - Arrête la réception de la position d'un axe par bus CAN

Syntaxe : STOPCANRECEIVEPOSITION (<PDO>)

Description : Cette instruction arrête la réception de la position d'un axe par bus CAN.

Valeurs acceptées : < PDO> numéro de PDO de 1 à 8

U) STOPCANSENDPOSTION - Arrête l'envoie de la position de l'axe sur le bus CAN

Syntaxe : STOPCANSENDPOSITION (<PDO>)

Description : Cette instruction arrête l'envoie de la position sur le bus CAN.

Valeurs acceptées : < PDO> numéro de PDO de 1 à 8

V) VF, VB, VI, VL et VR - Lecture ou écriture d'une variable distante

Syntaxe 1: VF (<Numéro Variable>, <Node>) = <bit ou variable>

Syntaxe 2: <Variable> = VF (<Numéro Variable>, <Node>)

Syntaxe 3: VB (<Numéro Variable>, <Node>) = <octet ou variable>

Syntaxe 4: <Variable> = VB (<Numéro Variable>, <Node>)

Syntaxe 5: VI (<Numéro Variable>, <Node>) = <entier ou variable>

Syntaxe 6: <Variable> = VI (<Numéro Variable>, <Node>)

Syntaxe 7: VL (<Numéro Variable>, <Node>) = <entier long ou variable>

Syntaxe 8: <Variable> = VL (<Numéro Variable>, <Node>)

Syntaxe 9: VR (<Numéro Variable>, <Node>) = <réel ou variable>

Syntaxe 10: <Variable> = VR (<Numéro Variable>, <Node>)

Limites : < Numéro Variable > : de 0 à 255

<Node> : de 0 à 255

Syntaxe 1 et 2 : <Variable> : de 0 à 1

Syntaxe 3 et 4 : <Variable> : de 00h à FFh

Syntaxe 5 et 6 : <Variable> : de 0000h à FFFFh

Syntaxe 7 et 8 : <Variable> : +/- 7FFFFFFFh

Syntaxe 9 et 10 : <Variable>: +/- 7FFFFFFFh

Description : Cette fonction permet de lire ou d'écrire une variable d'un drive du réseau CANopen.

10-2-4- Exemples

A) Echange de variables entre drive

a) Modification d'une variable d'un autre drive :

```
VR(2,3)=VR1 ‘Envoie le contenu de VR1
            ‘ dans la variable VR2 du drive n° 3
```

b) Lecture d'une liste de variable d'un autre drive :

```
VB1=0
REPEAT                          ‘ Lit dans le drive n°5
    VR0[VB1]= VR(VB1,5)         ‘ les variables  VR0 à VR9
    VB1=VB1+1                   ‘ et les écrit dans ce drive
UNTIL VB1=10
```

B) Echange par SDO

a) Lecture de l'état des entrées du drive IMD n°3

```
CANopenL(1280h,1)=603h 'Initialisation du TX client SDO
CANopenL(1280h,2)=583h 'Initialisation du RX Client SDO
BOUCLE:
    DELAY 10 ‘Tempo de 10ms
    Entrees = SDOI(60FDh,0) ‘Lecture des entrées du drive n°3 via SDO
GOTO BOUCLE
```

b) Ecriture des sorties du drive IMD n°5

```
CANopenL(1280h,1)=605h 'Initialisation du TX client SDO
CANopenL(1280h,2)=585h 'Initialisation du RX Client SDO
SDOI(60FEh,0) = 0 ‘Ecriture des sorties du drive n°5 via SDO
OldSorties = 0
BOUCLE:
```

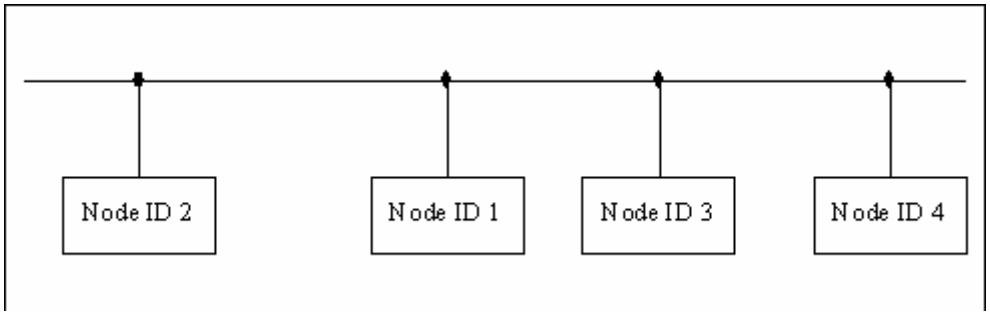
```
    IF OldSorties <> Sortie THEN
        SDOI(60FEh,0) = Sorties 'Ecriture des sorties du drive n°5 via SDO
        OldSorties = Sorties
    END IF
GOTO BOUCLE
```

C) Echange par PDO

La communication par PDO est différente dans l'esprit par rapport aux SDO précédemment décrits.

Là où les SDO envoient des données à un périphérique bien défini, le PDO envoie un octet sur le réseau, et tous les périphériques ayant été paramétrés en adéquation avec cette émission reçoivent cette information.

Il existe différents PDO. Pour l'exemple présenté ici, nous n'utiliserons que le premier PDO. Les autres fonctionnent exactement de la même façon.



Prenons l'exemple du réseau ci-dessus, avec différents périphériques. Chaque élément de ce réseau possède 2 paramètres importants relatifs à un PDO, un pour l'émission et un pour la réception. Ces paramètres sont des COB-ID.

Ce sont ces paramètres qui définissent à qui sont destinés les messages PDO envoyés sur le réseau par n'importe quel périphérique.

Pour chaque drive, les paramètres de transmission par défaut du premier PDO sont :

200h + Node-ID en réception

180h + Node-ID en émission

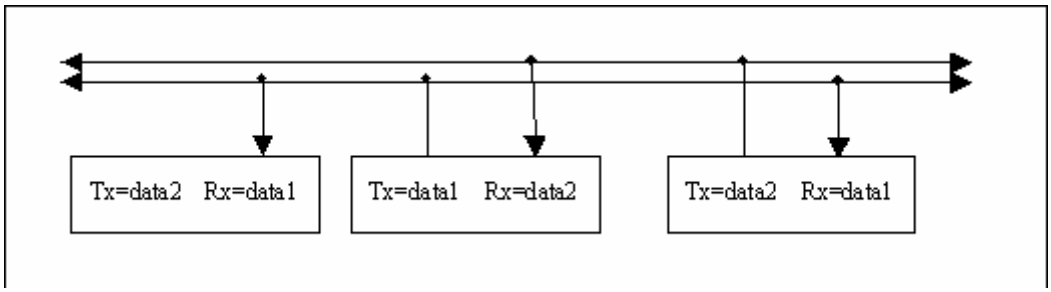
Prenons par exemple le cas du périphérique 2, valeur de Node-ID 2.

Ses paramètres par défaut seront 202h en réception et 182h en émission.

Cela signifie que, en gardant ces paramètres par défaut, lorsque ce périphérique émettra un PDO, celui-ci aura comme COB-ID 182h. Alors tous les périphériques dont le COB-ID de réception du PDO 1 seront accordés avec cette valeur 182h, recevront ce PDO.

De la même façon, le périphérique 2 recevra tous les PDO émis par les périphériques dont le COB-ID de PDO en émission sera égal à 202h.

On peut symboliser ce fonctionnement par le schéma suivant :



Donc, plusieurs paramétrages sont possibles pour l'envoi d'un PDO, la seule condition est que le COB-ID de transmission de l'émetteur soit égal au COB-ID de réception du récepteur.

a) Envoie de la position du drive n°0 dans le 4ième PDO de transmission :

```
' numéro d'identifiant COBID  en transmission : n°PDO – 8 – n° node
CANopenL(1803h,01h)=00000481h ‘ PDO n°4 – 8 – drive n°1
' Transmission type
CANopenB(1803h,02h)=0FFh ‘cycliquement
' Number of mapped PDO
CANopenB(1A03h,00h)=01h ‘nombre d'object du PDO
' PDO mapping
CANopenL(1A03h,01h)=64100120h ‘ mapping du 1er object
' Inhibit time
CANopenI(1803h,03h)=00h ‘temps de rafraîchissement en 100µs

' Démarre le CAN PDO
SetupCan(0,1)
Can(0)=2   ‘longueur du message
Can(1)=1   ‘NMTstart
Can(2)=0   ‘Node  : all
CanTx

lp:
```

```
    PDOTx(3) ’force l’envoie du PDO-1 sélectionné
goto lp
```

b) Réception la position du drive n°0 dans le 4ième PDO de réception :

```
CANopenL(1403h,01h)=00000481h
' Transmission type
CANopenB(1403h,02h)=0FFh ‘sur changement d’état
' Number of mapped PDO
CANopenB(1603h,00h)=01h ‘nombre d’object du PDO
' PDO mapping
CANopenL(1603h,01h)=34000020h ‘ mapping du 1er object VR0

' Démarre le CAN PDO
SetupCan(0,1)
Can(0)=2   ‘longueur du message
Can(1)=1   ‘NMTstart
Can(2)=0   ‘Node  : all
CanTx

lp:
    IF VR0 > 100 THEN
        OUT(2)= 1
    ELSE
        OUT(2)= 0
    END IF
goto lp
```

D) Exemple de CAN générique

```
SetupCan(1,1)
Can(0)=2
Can(1)=1
Can(2)=0
CanTx
VI5=CanErrCounter
VB5=CanErr
if CanEvent=0 Goto St
    VB0=Can(0)
    VB1=Can(1)
    VB2=Can(2)
    VB3=VB3+1
St:
if VF10=0 goto st2
    CanErrCounter=0
    CanErr=0
St2:
```

10-3- MODbus

10-3-1- Définition

A) Introduction

Le protocole MODBUS est un protocole maître/esclave utilisé principalement dans le milieu industriel. Il permet à des équipements de supervision (Human Machine Interface, Supervisory Control And Data Acquisition), de communiquer avec un ou plusieurs équipements industriels (Programmable Logic Controllers, automates, sondes, etc..).

Ce protocole fonctionne sous forme de requête. Ces messages transitent sur un support physique qui peut être une liaison asynchrone RS232, RS422 ou RS485.

Pour distinguer un équipement esclave d'un autre, on attribue un numéro d'identification (Unit ID) à chaque équipement. Grâce à ce numéro et dans le cas d'une liaison à plusieurs (cas du RS485) seul l'équipement esclave concerné répondra à une requête d'un équipement maître.

Le variateur gère le protocole MODBUS RTU Esclave.

Le format de la liaison est 8 bits de données, 1bit de stop et pas de parité.

La vitesse de transmission peut aller jusqu'à 57600 bauds

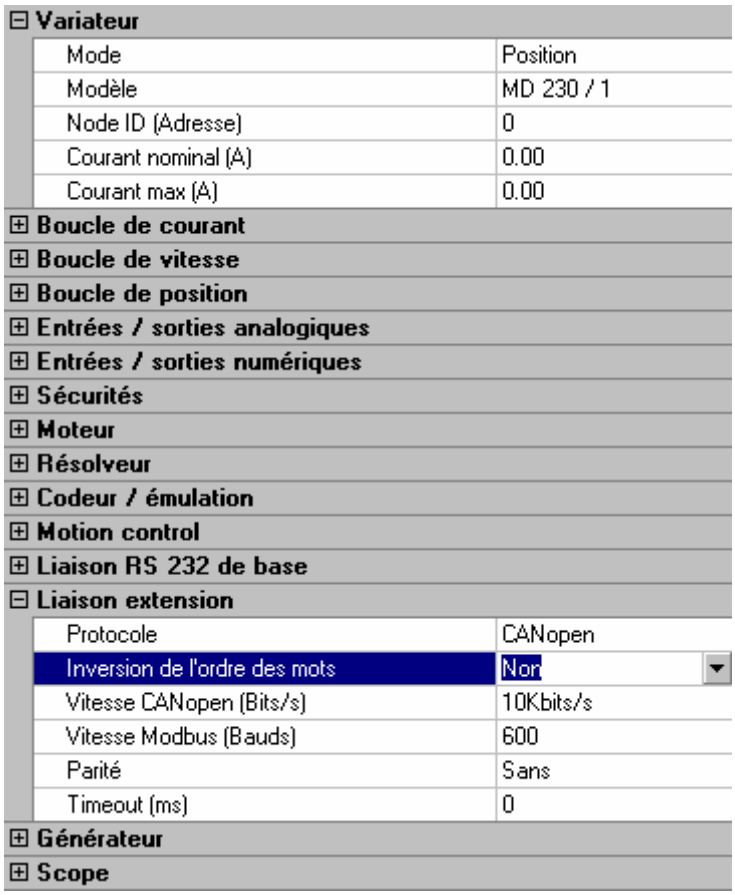
Les fonctions de lecture de mots (fonction n°3 ou 4) et écriture des mots (fonction n°16) sont reconnues par le variateur.

B) Variables codées sur 2 mots

Les paramètres du variateur ainsi que les variables de type entier long et réel sont codés sur 2 mots (32bits). Comme l'indique la norme Modbus, un double mot est de la forme suivant :

Adresse :	Mot :
n	Poids fort
n+1	Poids faible

Le paramètre « Inversion de l'ordre des mots » accessible à partir de la liste des paramètres dans le groupe Liaison extension permet d'inverser le codage du double mot sur les paramètres et les variables de type entier long et réel.



Type liaison system	Paramètre inversion	Format des données	Inversion Codage VR et VL	Inversion Codage paramètres
Activé	X	Forcé à float	Non	Non
Non activé	Non	Float ou Decimal	Non	Non
Non activé	Oui	Float ou Decimal	Oui	Oui

* X : état indifférent

Si Inversion codage = NON ⇒ Adresse n : poids fort

Adresse n+1 : poids faible

Si Inversion codage = OUI ⇒ Adresse n : poids faible

Adresse n+1 : poids faible

10-3-2- Dictionnaire

A) Dictionnaire du MODBus :

Le dictionnaire contient les différents paramètres et variables du variateur.

Voir l'écran Aide \ Modbus-CANopen

- Paramètres accessibles entre les adresses 1000 et 1400
- Variables type flag sont accessibles entres les adresses E000h et E00Fh
- Variables type octet sont accessibles entres les adresses E010h et E08Fh
- Variables type entier sont accessibles entres les adresses E090h et E18Fh
- Variables type entier long sont accessibles entres les adresses E190h et E38Fh
- Variables type réel sont accessibles entres les adresses E390h et E58Fh

Description des colonnes :

- Modbus : adresse Modbus du paramètre
- Name : nom interne de paramètre

- Description : description du paramètre
- Size : taille de paramètres en nombre de mots
- Type :
    - Fixed, l'unité est fixe
    - DPL, l'unité dépend du paramètre float/décimal (si décimal, la valeur dépend de la précision du DPL)
- Access : autorisation d'accès au paramètre à partir des taches DPL
- Read index : numéro d'index de lecture du paramètre
- Read subindex : numéro de sous index de lecture du paramètre
- Write index : numéro d'index d'écriture du paramètre
- Write subindex : numéro de sous index d'écriture du paramètre

Comparatif table modbus IMD et MD :

Adresse	MD	IMD	Commentaire
0x0000	Réservé	Réservé	
0x0258	Paramêtres	Réservé	
0x03E8	Réservé	Paramêtres	
0x2000	Réservé	FRAM	4kMot/accès direct
0x3000	Réservé	Réservé	
0x8000	Echange PC	Echange PC	Réservé
0xEFFF	Variables	Variables	voir dictionnaire modbus
0xFFFF			

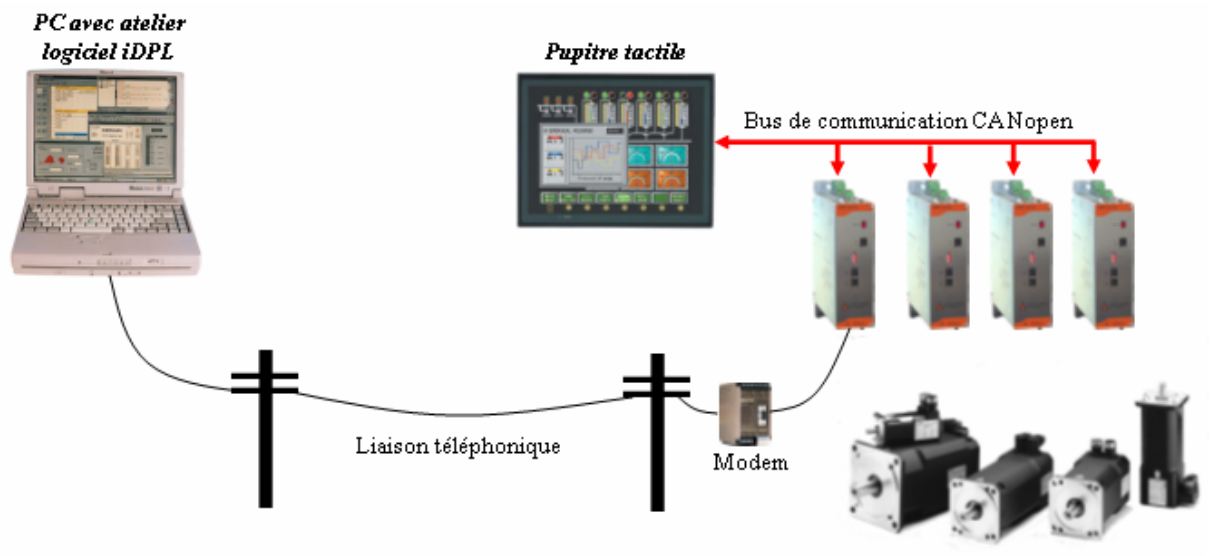
10-4- Télémaintenance

La télémaintenance permet par une liaison téléphonique de contrôler à distance un ou plusieurs IMD à l'aide du logiciel iDPL. La télémaintenance se compose d'un numéroteur téléphonique intégré au logiciel iDPL, de deux modems reliés entre eux par une ligne téléphonique.

10-4-1- Raccordement

A) Architecture

Les différents éléments sont connectés de la façon suivante :



B) Liaison RS 232 entre le modem 1 et le variateur

Brochage des connecteurs SUBD 9 points :

Pin	IMD	Modem
1		CD
2	RXD	RXD
3	TXD	TXD
4		DTR

5	GND	GND
6		DSR
7		RTS
8	CTS	CTS

Prévoir un câble blindé avec blindage relié aux deux extrémités.

C) Liaison RS 232 entre le modem 2 et le PC

La liaison entre le modem et le PC est réalisée directement par le câble fourni avec le modem.

10-4-2- Etablissement de la liaison

A) Paramétrage du modem 1 relié au variateur IMD

Le paramétrage du modem relié au variateur IMD s'effectue en reliant ce dernier à un PC. Pour ce faire on utilise un logiciel terminal pour envoyer les commandes au modem.

Ce paramétrage a pour but d'effectuer les opérations suivantes :

- Initialiser le modem
- Définir le nombre de sonnerie avant le décrochage du modem pour permettre l'établissement automatique de la liaison.
- Supprimer les contrôles de flux matériel et logiciel.
- Stocker cette configuration dans la mémoire non volatile du modem
- Définir ces paramètres en mémoire non volatile comme paramètres à utiliser à la mise sous tension.

Exemple de paramétrage d'un modem de type « 3Com Us Robotics Sportster » :

- Commande : AT&F0

Signification : Chargement des paramètres d'usine.

- Commande : ATS0=3

Signification : Décrochage automatique au bout de 3 sonneries.

- Commande : AT&H0

Signification : Désactive le contrôle de flux en émission

- Commande : AT&I0

Signification : Désactive le contrôle de flux en réception

- Commande : AT&W0

Signification : Stockage des paramètres courant dans la mémoire non volatile N°0

- Commande : ATY0

Signification : Définir les paramètres en mémoire non volatile N°0 comme paramètres à utiliser à la mise sous tension.

Lorsque ces commandes sont prise en compte par le modem celui-ci répond « OK » .

Paramétrage d'un modem de type « Westermo TD31 ou TD32 » :

- Commande : AT&F

Signification : Chargement des paramètres d'usine.

- Commande : ATS0=3

Signification : Décrochage automatique au bout de 3 sonneries.

- Commande : AT&C1

Signification : Active DCD à la connexion

- Commande : AT&K0

Signification : Désactive le contrôle de flux

- Commande : AT&W0

Signification : Stockage des paramètres courant dans la mémoire non volatile N°0

- Commande : AT&Y0

Signification : Définir les paramètres en mémoire non volatile N°0 comme paramètres à utiliser à la mise sous tension.

Lorsque ces commandes sont prise en compte par le modem celui-ci répond « OK » .

B) Paramétrage du modem 2 relié au PC

Le paramétrage du modem relié à la PC s'effectue à la rubrique « Modem » du fichier MCB.INI se trouvant dans le répertoire Data du iDPL.

Ce paramétrage a pour but d'effectuer les opérations suivantes :

- Initialiser le modem
- Supprimer la prise en compte des signaux DSR et DTR pour éviter un raccrochage automatique en cas de fermeture du port de communication.
- Définir la méthode d'appel et de raccrochage du modem.
- Définir les messages renvoyés par le modem.
- Les paramètres sont initialisés automatiquement à des valeurs par défauts permettant de fonctionner avec les modems courants.

Exemple de paramétrage d'un modem de type « 3Com Us Robotics Sportster » :

- Paramètre : Init1

Valeur : ATZ

Signification : Chargement des paramètres d'usine.

- Paramètre : Init1TimeOut

Valeur : 5

Signification : Délai en 1/10 de seconde d'attente maxi de la réponse du modem.

- Paramètre : Init2

Valeur : AT&D0&S0

Signification : Suppression de la prise en compte de DTR et DSR

- Paramètre : Init2TimeOut

Valeur : 5

Signification : Délai en 1/10 de seconde d'attente maxi de la réponse du modem.

- Paramètre : Dial

Valeur : ATDT pour numérotation vocale. ATDP pour numérotation impulsionnelle

Signification : Définition de la méthode d'appel.

- Paramètre : DialTimeOut

Valeur : 600

Signification : Délai en 1/10 de seconde d'attente maxi avant la connexion.

- Paramètre : Ok

Valeur : OK

Signification : Réponse du modem si la commande est exécutée correctement.

- Paramètre : Connect

Valeur : CONNECT

Signification : Définir le message renvoyé par le modem à la connexion.

- Paramètre : Busy

Valeur : BUSY

Signification : Définir le message renvoyé par le modem si la ligne est occupée.

- Paramètre : Hangup

Valeur : ATH

Signification : Définition de la méthode de raccrochage

- Paramètre : HangupOk

Valeur : NO CARRIER

Signification : Définir le message renvoyé par le modem lorsqu'il raccroche la ligne

- Paramètre : CommandTimeOut

Valeur : 20

Signification : Délai en 1/10 de seconde d'attente maxi avant le passage en mode commande.

- Paramètre : HangupTimeOut

Valeur : 20

Signification : Délai en 1/10 de seconde d'attente maxi avant le raccrochage.

Ces paramètres sont automatiquement fixés aux valeurs par défauts indiquées lors de la première utilisation.

Paramétrage d'un modem de type « Westermo TD31 ou TD32» :

- Paramètre : Init1

Valeur : ATZ

Signification : Chargement des paramètres d'usine.

- Paramètre : Init1TimeOut

Valeur : 20

Signification : Délai en 1/10 de seconde d'attente maxi de la réponse du modem.

- Paramètre : Init2

Valeur : AT&F&K0

Signification : Suppression de la prise en compte de DTR et DSR

- Paramètre : Init2TimeOut

Valeur : 20

Signification : Délai en 1/10 de seconde d'attente maxi de la réponse du modem.

- Paramètre : Dial

Valeur : ATDT pour numérotation vocale. ATDP pour numérotation impulsionnelle

Signification : Définition de la méthode d'appel.

- Paramètre : DialTimeOut

Valeur : 600

Signification : Délai en 1/10 de seconde d'attente maxi avant la connexion.

- Paramètre : Ok

Valeur : OK

Signification : Réponse du modem si la commande est exécutée correctement.

- Paramètre : Connect

Valeur : CONNECT

Signification : Définir le message renvoyé par le modem à la connexion.

- Paramètre : Busy

Valeur : BUSY

Signification : Définir le message renvoyé par le modem si la ligne est occupée.

- Paramètre : Hangup

Valeur : ATH

Signification : Définition de la méthode de raccrochage

- Paramètre : HangupOk

Valeur : NO CARRIER

Signification : Définir le message renvoyé par le modem lorsqu'il raccroche la ligne

- Paramètre : CommandTimeOut

Valeur : 20

Signification : Délai en 1/10 de seconde d'attente maxi avant le passage en mode commande.

- Paramètre : HangupTimeOut

Valeur : 20

Signification : Délai en 1/10 de seconde d'attente maxi avant le raccrochage.

Le numéroteur téléphonique suppose que le modem est paramètré pour recevoir un écho aux commandes envoyées et un message texte en réponse. Dans le cas contraire la communication serait impossible. Il est possible de s'assurer du bon fonctionnement du modem en le configurant avec les paramètres usine par défaut.

Pour ce faire on utilise un logiciel terminal pour envoyer les commandes au modem.

Paramétrage d'un modem de type « 3Com Us Robotics Sportster » :

- Commande : AT&F

Signification : Chargement des paramètres d'usine.

- Commande : AT&W0

Signification : Stockage des paramètres courant dans la mémoire non volatile N°0

- Commande : ATY0

Signification : Définir les paramètres en mémoire non volatile N°0 comme paramètres à utiliser à la mise sous tension.

Paramétrage d'un modem de type « Westermo TD31 ou TD32» :

- Commande : AT&F

Signification : Chargement des paramètres d'usine.

- Commande : AT&W0

Signification : Stockage des paramètres courant dans la mémoire non volatile N°0

- Commande : AT&Y0

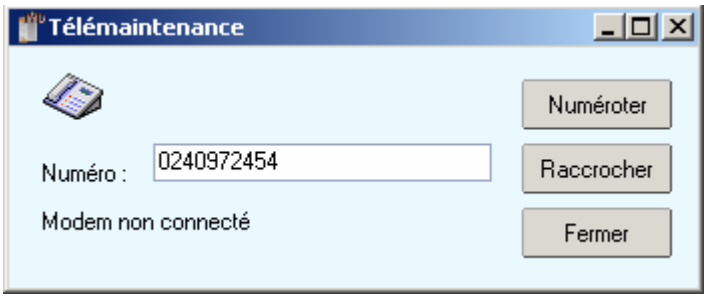
Signification : Définir les paramètres en mémoire non volatile N°0 comme paramètres à utiliser à la mise sous tension.

ATTENTION :

- Pour les modems Westermo, il est également recommandé de laisser la configuration des Dips par défaut (tous sur OFF).

C) Appel

A partir du numéroteur téléphonique intégré au logiciel iDPL, on peut établir et interrompre la liaison téléphonique. Le numéroteur téléphonique est accessible à partir du menu communication / télémaintenance.



Après la saisie du numéro de téléphone, appuyer sur le bouton «Numéroter» pour établir la liaison. Le bouton «Raccrocher » permet quant a lui d'arrêter la liaison.

Lorsque la liaison est établie, on peut utiliser toutes les fonctions du iDPL :

- Envoyer et recevoir la configuration, les variables, les tâches, les cames, la mémoire FRAM etc ...
- Démarrer et arrêter les tâches
- Accès aux outils de debug : Hyperterminal, Scope, Tableau de bord.
- Recharger l'operating system.
- Accéder aux autres drives du réseau CANopen et effectuer toutes ces opérations

10-4-3- Liste des modems validés

3 Com / US Robotics

- Sportster Voice 33600 Fax Modem
- Sportster 56 K Fax Modem

Westermo

- TD 31
- TD 32

Index

A

ACC 208, 209
ACC% 209
Affectation et brochages des connecteurs 19
Aide 90
AND 210
Arbre électrique 158, 159
ARCCOS 210
ARCSIN 210
ARCTAN 211
Arrêt d'un mouvement 156
Attente active 191
Attente d'un état 189
Attente passive 191
Auto tuning des boucles de régulations 95
AXIS 211
AXIS_S 212

B

Boîte à cames 196

C

Câbles 28
Call 212, 213
CAMBOX 213
CAMBOXSEG 213
Came 168, 169, 170, 171, 172, 173, 174, 175, 176
CAMNUM_S 214
CAMREADPOINT 214
CAMSEG_S 214
CAN 277
CANERRCOUNTER 278
CANEVENT 278
CANPOSTIMEOUTRAZ 279
CANSENDNMT 279
CANSENDSYNCHRO 280
CANSETUPSYNCHRO 280
Capture 180, 181
CAPTURE1 215
Caractéristiques 272
CLEAR 216
CLEARMASTER 216
Communication 69
Compteurs 193
Configuration du réseaux 270
Contenu d'un projet 46
CONTINUE 217
Conversions de type de données 127
COS 217
COUNTER_S 218

D

DAC 218

DEC 219
DEC% 219
Déclaration d'un axe en mode virtuel 152
DELAY 219, 220
Description 261
Diagnostic du bus 274
Dictionnaire 276, 290
DISABLERECALE 220
DISPLAY 220

E

Echange par SDO 284
Echange par PDO 285
Echanges de variables entre drive 284
Ecran initial 39
Ecriture des sorties 188
Ecriture d'une sortie 190
ENABLERECALE 220, 221
ENDCAM 222
Exemple de CAN générique 287
EXIT SUB 221
EXP 222

F

FE_S 223
FEMAX_S 222, 223
FILTERMASTER 223
Fonctionnement 109, 113

G

GEARBOX 224
GEARBOXRATIO 225
Généralités 13
Gestion d un projet 42
Gestion des tâches 134
Goto 225

H

HALT 225, 226
HOME 226, 227
HOME_S 227
HOMEMASTER 227, 228
HOMEMASTER_S 228

I

ICORRECTION 229
IF 229, 230
INP 230
INPB 230
INPW 231
INT 231
Introduction 108, 124, 142, 267, 288

L

La communication CANopen 268
Langage DPL 86
Lecture des entrées 188
Lecture des sorties 189
Lecture d'une entrée 190

Les données sauvegardées 129
Les modes d utilisation 36
Les modes de fonctionnement 94
Les paramètres 132, 133
Liste des instructions CANopen 276
LOADCAM 231
LOADPARAM 233
LOADTIMER 233
LOG 233
LOOP 234

M

Maître virtuel 186
MASTEROFFSET 234
MERGE 234
Messages afficheur STATUS 7 segments 263
Mise en oeuvre 114
Mise en place 110, 122
MOD 235
Montage 18
Motion control 81
Mouvement de correction 166
Mouvements absolus 153
Mouvements déclenchés 183
Mouvements infinis 155
Mouvements relatifs 154
Mouvements synchronisés 161
MOVA 235
MOVE_S 235, 236
MOVEMASTER_S 236
MOVR 236, 237
MOVS 237
Multiaxes par CANopen 176

N

Nexttask 237
NOT 237
Notation numériques 128

O

Options 89
OR 238
ORDER 238
ORDER_S 238
OUT 239
OUTB 239
Outils de réglages 73

P

Paramétrage d'un axe 142
Paramètres 51, 52, 53, 56, 57, 58, 60
Passage en mode asservi 146
Passage en mode non asservi 145
PDOEVENT 280
PDOTX 280
POS 239, 240
POS_S 240
Présentation_de_la_carte_IMDCANI 272
Principes du multitâches 134

Priorité des tâches....134
Procédure d'installation du logiciel DPL....35
Profil de vitesse....145
Projet....48, 49

R

Raccordement....272
READI....241
READL....242
READPARAM....242
READR....242
Recalage automatique....183
Réglage de la boucle de courant....96, 97
Réglage de la boucle de position....102, 103
Réglage de la oucle de vitesse....99
Réglage des paramètres moteur et résolveur....91
Réglage du déverrouillage variateur....93
Réglage en boucle de vitesse....106
Réglage en double boucle résolveur/codeur....107
Réglage en entrée stepper....107
RESTART....244
RUN....244

S

SAVEPARAM....244
SAVEVARIABLE....245
Schémas de raccordement....30
SECURITY....245
SETUPCAN....281
SGN....246
SIN....246
SLAVEOFFSET....246
SQR....247
SSTOP....247
SSTOPMASTER....247, 248
STARTCAM....248
STARTCANRECEIVEPOSITION....282
STARTCANSENDPOSITION....282
STATUS....249
STOP....249, 250
STOPCANRECEIVEPOSITION....283
STOPMASTER....251
STOPS....250
STOPS_S....250
Structure d'une tâche basic....136
STTA....251
STTI....252
STTR....252
SUSPEND....253

T

TAN....253
Test....198
Test d'un état....190
TIME....253, 254
TRAJA....254
TRAJR....255
TRIGGERC....255, 256
TRIGGERI....256
TRIGGERP....256

Type de messages envoyés....271

U

Unité utilisateur....144

V

Variables globales....126
Variables globales sauvegardées....128
Variables utilisant 2 mots....289
VEL....257
VEL%....258
VEL_S....257
VELMASTER_S....258
Vérifications avant mise en route....33
VERSION....258
VIRTUALMASTER....258
Vue de dessous....17

W

WAIT....259
WRITECAM....259
WRITEI....259
WRITEL....259
WRITER....260

X

XOR....260